

2025년 상반기 K-디지털 트레이닝

# 변수와 타입

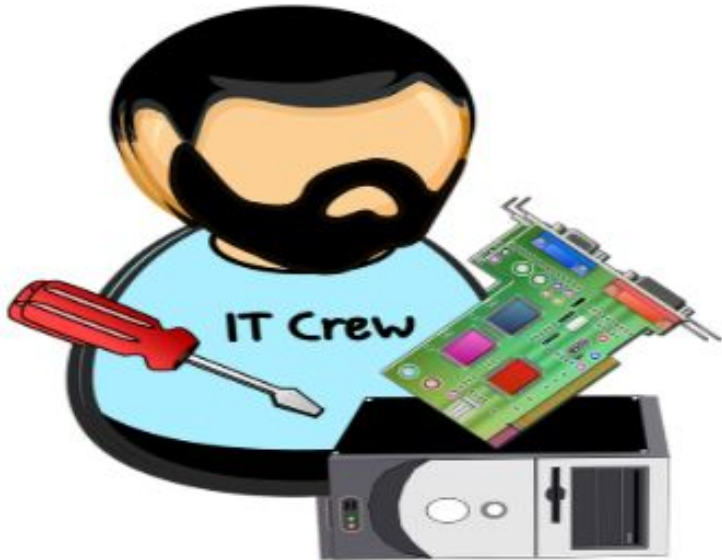
---

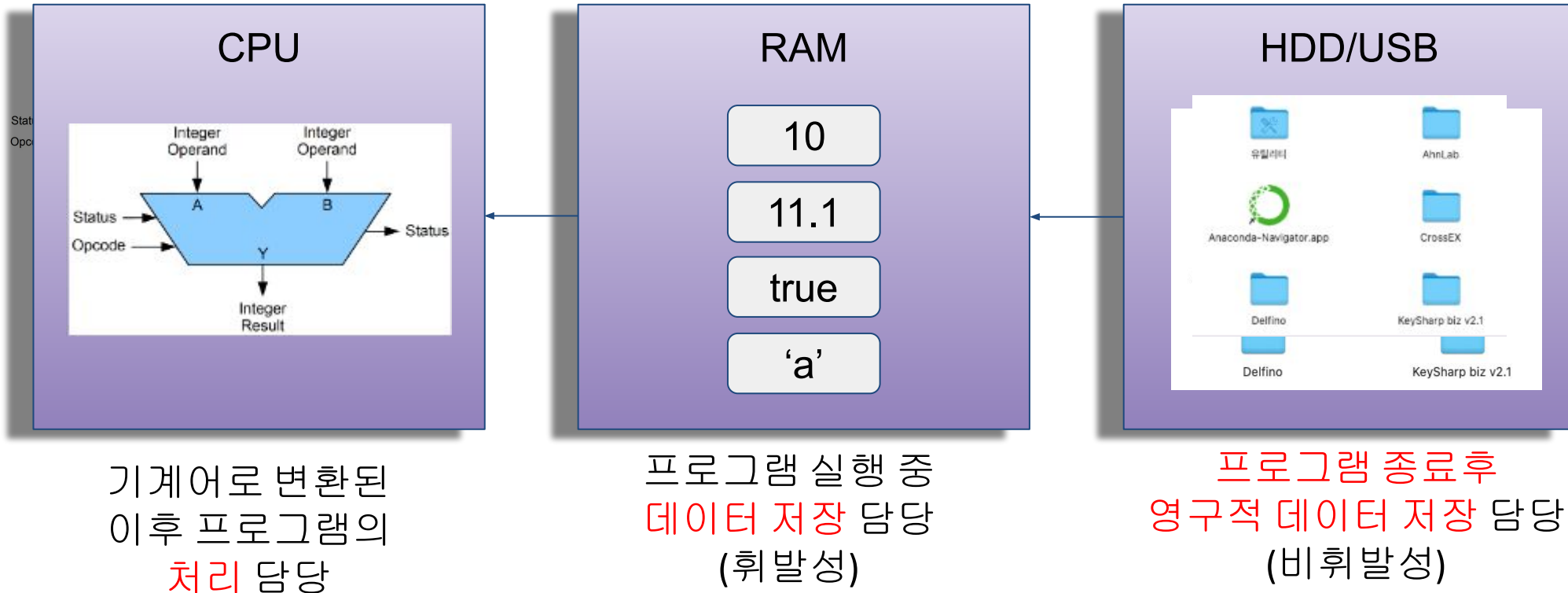
[KB] IT's Your Life

# know where, know who!

여러 개의 부품을 조립하듯이 코딩

-> 내가 원하는 부품을 얻는 것이 중요





- Java 소스코드 (.java) –컴파일 (javac)→ 바이트코드 (.class) –JVM 실행(java) → 기계어로 변환 → CPU가 명령어 실행
- JVM내에 있는 JIT컴파일러에 의해 실행최적화, 앞서 기계어로 변환된 것은 다시 변환하지 않음.

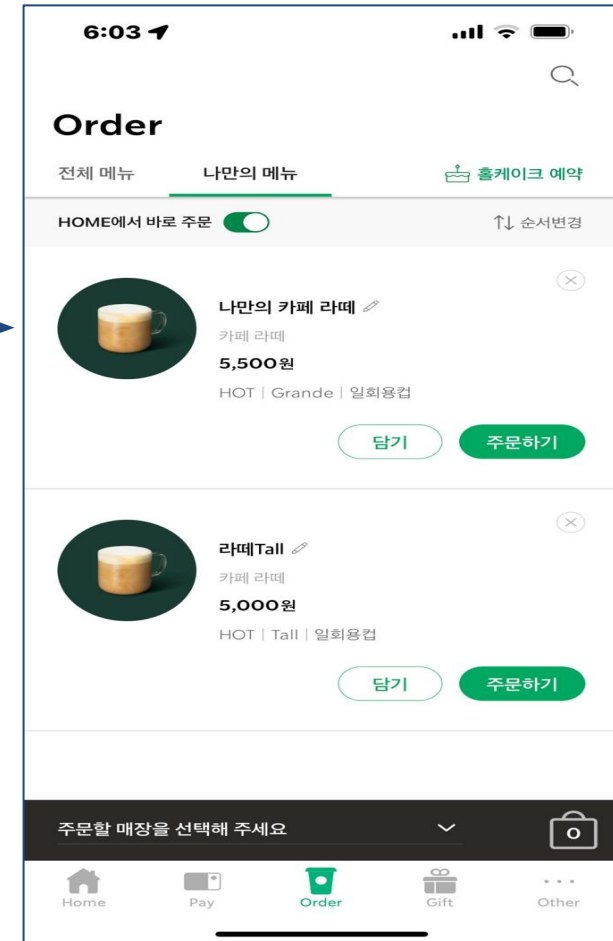
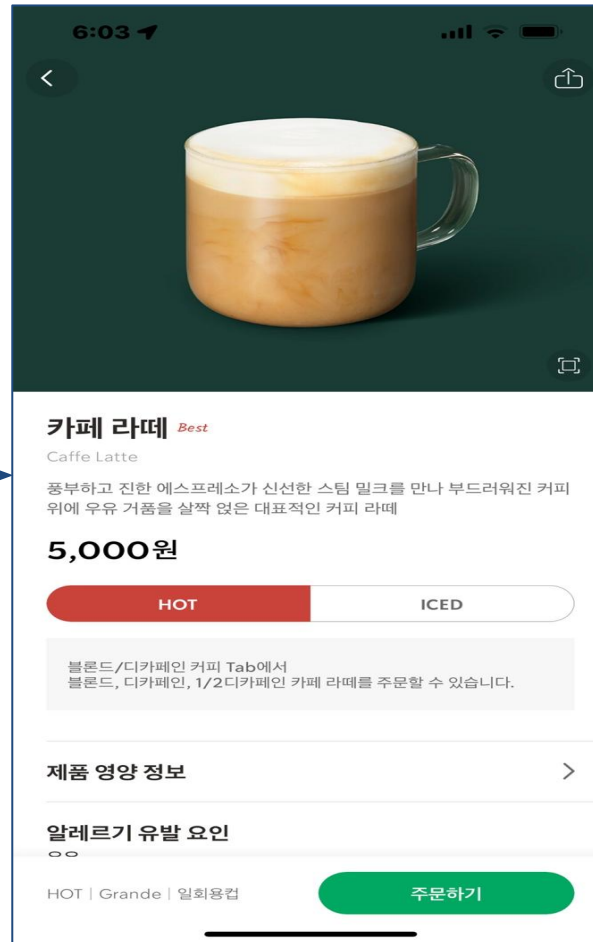


데이터를 ram에 넣어두었다가 cpu가  
꺼내어 처리하면서  
프로그램은 실행된다.

**어떤 데이터가 있는 것일까?**

프로그래밍을 하기 위해서는  
**사용할 데이터를 먼저 정의**해야  
한다.

- 시작하는 화면에 main()을 넣어주면, 여기서 부터 시작함. **프로젝트에서 main()은 하나**
- 다른 클래스의 메소드(함수)를 호출하는 식으로 프로그래밍이 구성



기준	JavaScript	Java
기본 개념	인터프리터 언어, 주로 웹 개발에 사용	컴파일 언어, 주로 서버 및 엔터프라이즈 애플리케이션에 사용
실행 환경	브라우저, Node.js	JVM (Java Virtual Machine)
언어 유형	동적 타이핑(dynamic typing)	정적 타이핑(static typing)
사용 사례	웹 프론트엔드 개발, 백엔드 개발 (Node.js), 모바일 앱 (React Native 등)	웹 백엔드, 안드로이드 앱 개발, 엔터프라이즈 애플리케이션, 데스크톱 애플리케이션
구문 및 문법	느슨한 문법, 함수형 프로그래밍 지원	엄격한 문법, 객체지향 프로그래밍
변수 선언	var, let, const	int, double, boolean, String, class 타입 객체
객체 지향	프로토타입 기반, class(ES6)	클래스 기반
멀티스레딩	이벤트 루프를 통한 비동기 처리 (단일 스레드)	멀티스레딩 지원 (다중 스레드)
패키지 관리	npm (Node Package Manager)	Maven Central, Gradle Repository
빌드 도구	Webpack, Rollup, Parcel, Vite	Maven, Gradle, Ant

변수에 들어가는 데이터의  
유형이 언제 결정되는가?  
JavaScript는 실행 중  
타입이 결정되고, Java는  
컴파일 시 결정  
(타입변경불가)

# 변수(variable, ram내의 저장공간)

## RAM ( Random Access Memory)



<변수명>

name

age

height

<데이터>

"Hello Java"

20

180

JVM

Runtime Data Area

Method  
Area

Stack

Heap



check point!!! 반드시 알아두자!!!!

=> 기본형은 정수, 실수, 문자, 논리

=> 나머지는 부품을 사용

데이터종류(변수 종류)  
자바 변수의 기본형(4가지)  
숫자형

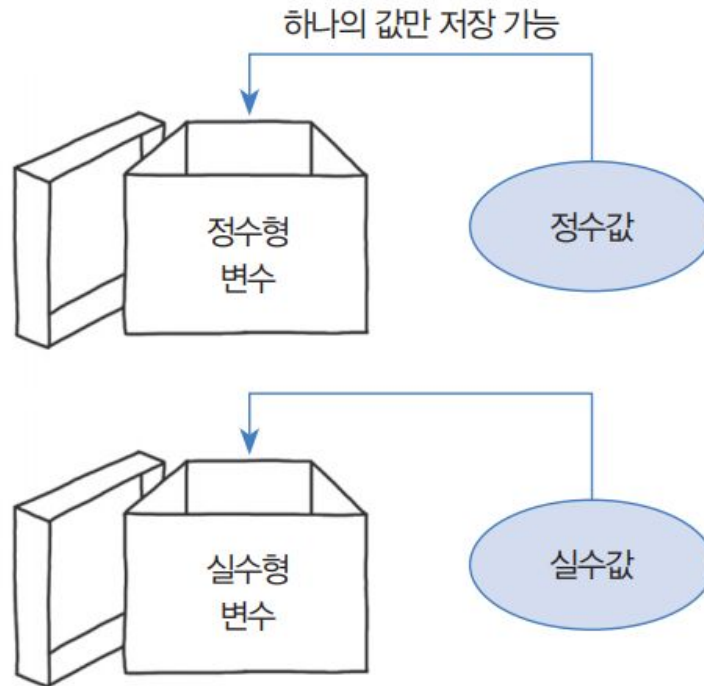
문자형  
논리형

정수형-소수점이 아닌 숫자  
실수형-소수점이 들어간 숫자  
한글자(a, A, 김,...)  
true(참)/false(거짓)

# 1 변수 선언

## • 변수 variable

- 하나의 값을 저장할 수 있는 메모리 번지에 붙여진 이름
- 선언된 타입의 값만 저장 가능 (정적타입핑)
  - 한 변수에는 선언된 타입과 일치하는 값만 저장 가능



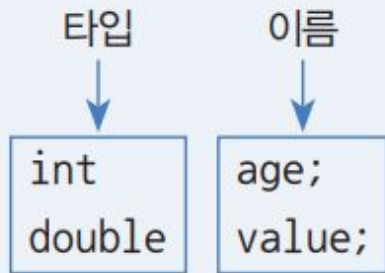
처음에 정수형  
변수로 선언된  
변수에는 정수형  
타입 값만 넣을 수  
있음.



# 1 변수 선언

## • 변수 선언

- 변수 변수를 사용하려면 변수 선언이 필요.
- 변수 선언은 저장할 데이터의 타입과, 변수 이름을 결정하는 것



//정수(int) 값을 저장할 수 있는 age 변수 선언

//실수(double) 값을 저장할 수 있는 value 변수 선언

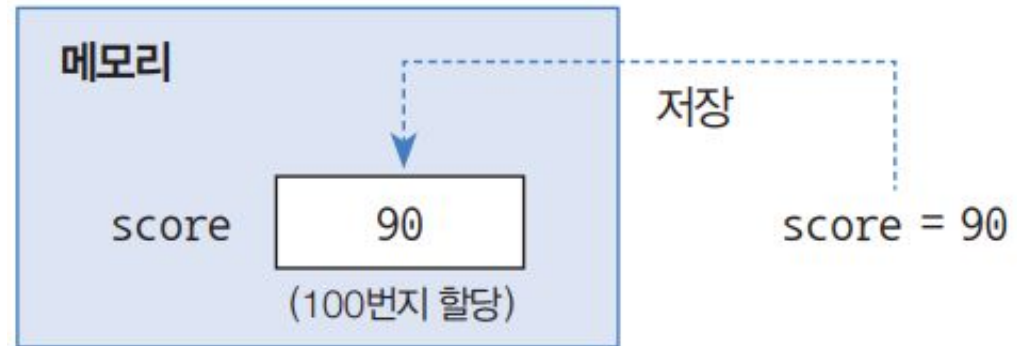
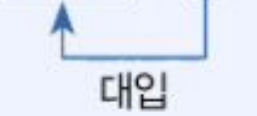
# 1 변수 선언

## • 변수 선언

### ○ 변수의 초기화

- 변수에 최초로 값이 대입될 때 메모리에 할당 되고, 해당 메모리에 값이 저장 → 초기화

```
int score;      //변수 선언
score = 90;     //값 대입
```



- 선언과 초기 값을 동시에 표현 가능

```
int score = 90;
```



# 1 변수 선언

- 변수 선언

- 변수의 초기화

- 초기화 되지 않은 변수를 사용(읽기)하면 컴파일 에러







```
① int value;           //변수 value 선언
② int result = value + 10;  //변수 value 값을 읽고 10을 더해서 변수 result에 저장
```

```
int value = 30;          //변수 value가 30으로 초기화됨
int result = value + 10;  //변수 value 값(30)을 읽고 10을 더해서 변수 result에 저장
```

## 변수(variable, ram내의 저장공간)

- 처음에 변수 정리되지 않은 상태로 저장공간이 만들어진다.
- “쓰레기값이 들어있다”라고 표현,
- 쓰레기값이 들어있는 변수는 프린트, 연산 불가능
- 변수에 어떤 값을 처음넣는 것을 “변수를 초기화한다”라고 표현

- 변수초기화하지 않은 경우
  - 1) 자바스크립트는 undefined
  - 2) 자바는 쓰레기값

변수 만들기	<code>int age;</code>	age 
변수에 값 저장	<code>age = 25;</code>	 → 
변수 만들면서 값 저장	<code>int age = 25;</code>	age 
변수값 변경	<code>age = 100;</code>	 → 

## 1 변수 선언

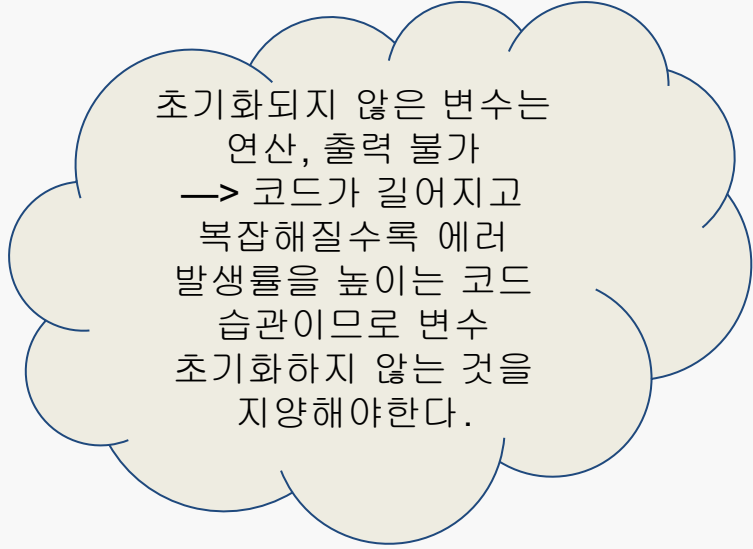
## • ch02.sec01.VariableInitializationExample.java

```
package ch02.sec01;

public class VariableInitializationExample {
    public static void main(String[] args) {
        //변수 value 선언
        int value; //초기화를 반드시 해주는 습관을 가져야한다!

        //연산 결과를 변수 result의 초기값으로 대입
        int result = value + 10;           //에러

        //변수 result 값을 읽고 콘솔에 출력
        System.out.println(result);       //에러
    }
}
```



초기화되지 않은 변수는  
연산, 출력 불가  
→ 코드가 길어지고  
복잡해질수록 에러  
발생률을 높이는 코드  
습관이므로 변수  
초기화하지 않는 것을  
지양해야한다.

# 1 변수 선언

- **ch02.sec01.VariableUseExample.java**

```
package ch02.sec01;  
  
public class VariableUseExample {  
    public static void main(String[] args) {  
        int hour = 3;  
        int minute = 5;  
        System.out.println(hour + "시간 " + minute + "분");  
  
        int totalMinute = (hour * 60) + minute;  
        System.out.println("총 " + totalMinute + "분");  
    }  
}
```

3시간 5분  
총 185분

## 1 변수 선언

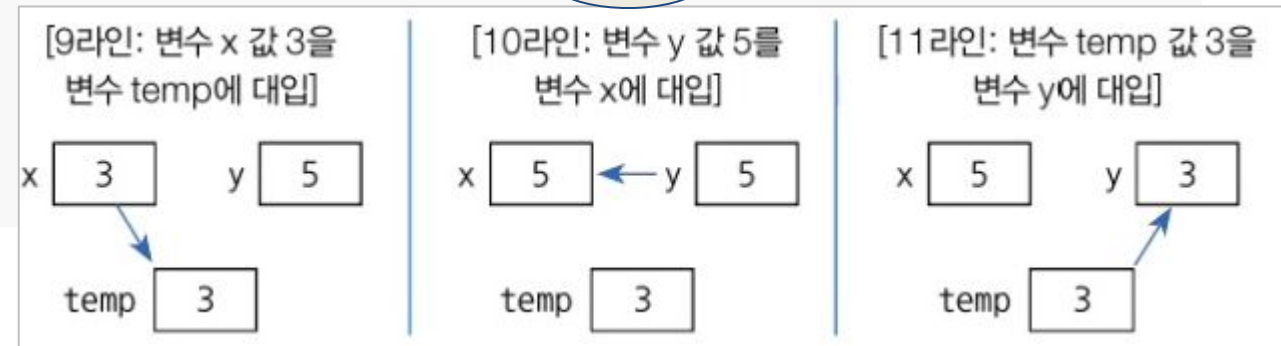
## • ch02.sec01.VariableExchangeExample.java

```
package ch02.sec01;
```

```
public class VariableExchangeExample {  
    public static void main(String[] args) {  
        int x = 3;  
        int y = 5;  
        System.out.println("x:" + x + ", y:" + y);  
  
        int temp = x;  
        x = y;  
        y = temp;  
        System.out.println("x:" + x + ", y:" + y);  
    }  
}
```

```
x:3, y:5  
x:5, y:3
```

순서대로 실행  
swap하는 예제



- 자바의 데이터 타입

- 기본 타입 **primitive type**(변수에 하나의 값이 저장)

값의 분류	기본 타입
정수	byte, char, short, int, long
실수	float, double
논리(true/false)	boolean

- 변수는 선언될 때의 타입에 따라 저장할 수 있는 **값의 종류와 허용 범위가 달라짐**  
→ 변수 선언이 필요한 이유



- byte, short, char, int, long 타입

- 정수 타입별 값의 범위

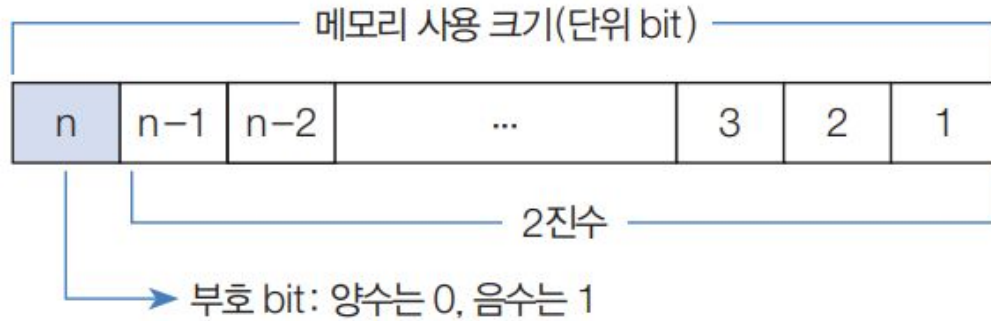
종류	byte	short	int	long
메모리 사용 크기(단위 bit)	8	16	32	64

타입	메모리 크기		저장되는 값의 허용 범위	
byte	1byte*	8bit	$-2^7 \sim (2^7-1)$	-128 ~ 127
short	2byte	16bit	$-2^{15} \sim (2^{15}-1)$	-32,768 ~ 32,767
char	2byte	16bit	$0 \sim (2^{16}-1)$	0 ~ 65535 (유니코드)
int	4byte	32bit	$-2^{31} \sim (2^{31}-1)$	-2,147,483,648 ~ 2,147,483,647
long	8byte	64bit	$-2^{63} \sim (2^{63}-1)$	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807

\* 1byte = 8bit, bit는 0과 1이 저장되는 단위

- **byte, short, char, int, long** 타입

- 메모리 크기를  $n$ 이라고 했을 때 정수 타입은 동일한 구조의 2진수로 저장

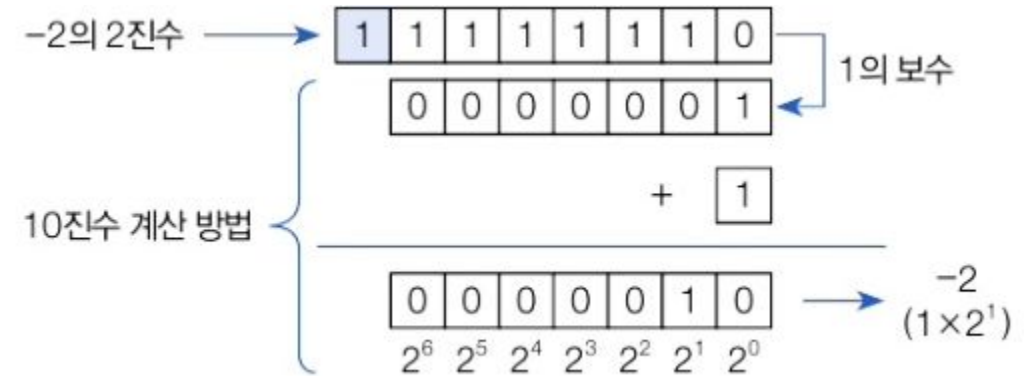
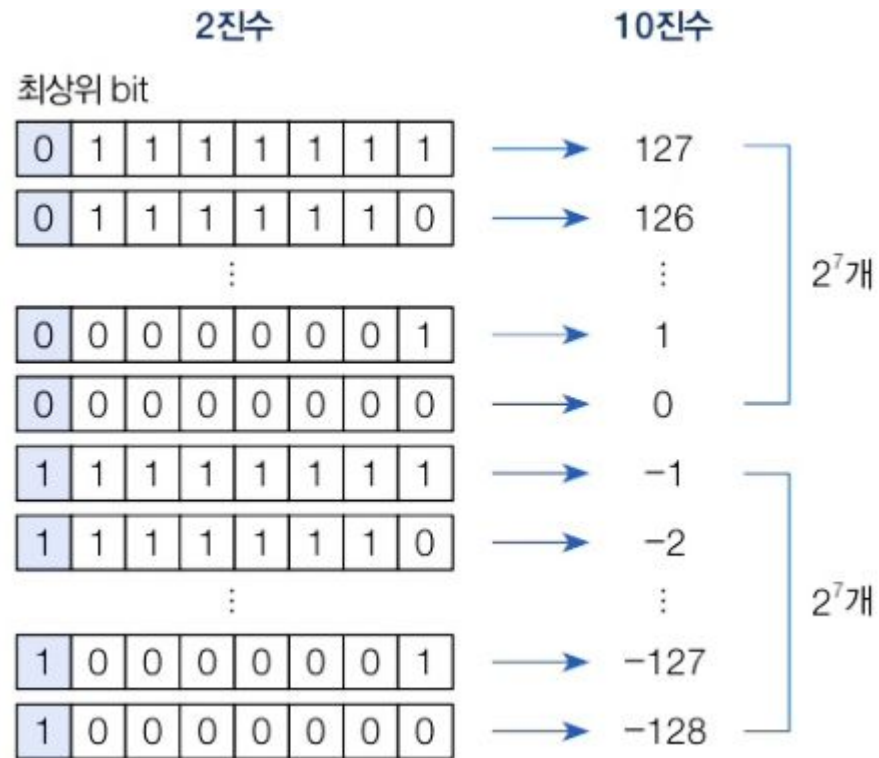


최상위 bit 값		값의 범위
0	$n-1$ 개의 bit	$0 \sim (2^{n-1} - 1)$
1	$n-1$ 개의 bit	$-2^{n-1} \sim -1$

- byte, short, char, int, long 타입

- 정수 표현 방법

- 최상위 비트는 부호( sign) 비트
- 음수는 2의 보수로 표현



- 정수 리터럴 표기법

정수는 2진수, 8진수,  
10진수, 16진수가 있음.

- 2진수: 0b 또는 0B로 시작하고 0과 1로 작성

```
int x = 0b1011;    //10진수 값 =  $1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 11$   
int y = 0B10100;  //10진수 값 =  $1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 20$ 
```

- 8진수: 0으로 시작하고 0~7 숫자로 작성

```
int x = 013;       //10진수 값 =  $1 \times 8^1 + 3 \times 8^0 = 11$   
int y = 0206;     //10진수 값 =  $2 \times 8^2 + 0 \times 8^1 + 6 \times 8^0 = 134$ 
```

- 정수 리터럴 표기법

- 10진수: 소수점이 없는 0~9 숫자로 작성

```
int x = 12;  
int y = 365;
```

- 16진수: 0x 또는 0X로 시작하고 0~9 숫자나 A, B, C, D, E, F 또는 a, b, c, d, e, f로 작성

```
int x = 0xB3;           //10진수 값 =  $11 \times 16^1 + 3 \times 16^0 = 179$   
int y = 0x2A0F;        //10진수 값 =  $2 \times 16^3 + 10 \times 16^2 + 0 \times 16^1 + 15 \times 16^0 = 10767$ 
```

- **ch02.sec02.IntegerLiteralExample.java**

```
package ch02.sec02;

public class IntegerLiteralExample {
    public static void main(String[] args) {
        int var1 = 0b1011; //2진수
        int var2 = 0206; //8진수
        int var3 = 365; //10진수
        int var4 = 0xB3; //16진수

        System.out.println("var1: " + var1);
        System.out.println("var2: " + var2);
        System.out.println("var3: " + var3);
        System.out.println("var4: " + var4);
    }
}
```

```
var1: 11
var2: 134
var3: 365
var4: 179
```

- ch02.sec02.ByteExample.java

```
package ch02.sec02;
```

```
public class ByteExample {  
    public static void main(String[] args) {  
        byte var1 = -128;  
        byte var2 = -30;  
        byte var3 = 0;  
        byte var4 = 30;  
        byte var5 = 127;  
        //byte var6 = 128; //컴파일 에러(Type mismatch: cannot convert from int byte)  
  
        System.out.println(var1);  
        System.out.println(var2);  
        System.out.println(var3);  
        System.out.println(var4);  
        System.out.println(var5);  
    }  
}
```

byte타입은 양수 127까지만  
넣을 수 있음.  
—> 해당 타입에 초과하는  
값을 넣는 경우에는 번역시  
(컴파일시) 에러남.

```
-128  
-30  
0  
30  
127
```

- ch02.sec02.LongExample.java

```
package ch02.sec02;
```

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

```
        long var1 = 10;
```

```
        long var2 = 20L;
```

```
        //long var3 = 1000000000000000; //컴파일러는 int로 간주하기 때문에 에러 발생
```

```
        long var4 = 1000000000000000L;
```

```
        System.out.println(var1);
```

```
        System.out.println(var2);
```

```
        System.out.println(var4);
```

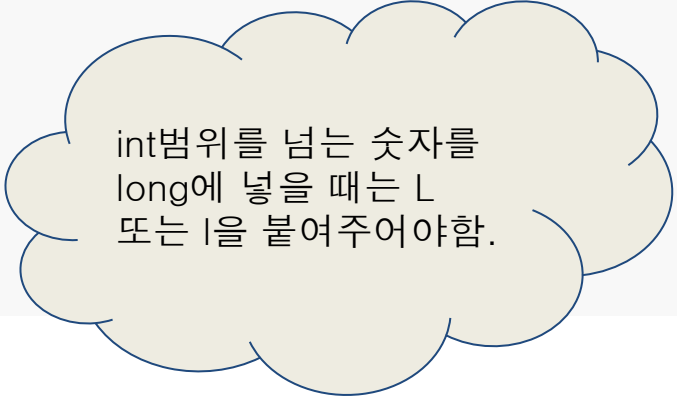
```
    }
```

```
}
```

```
10
```

```
20
```

```
1000000000000000
```



int범위를 넘는 숫자를  
long에 넣을 때는 L  
또는 l을 붙여주어야함.



## 3 문자 타입

### • 문자 리터럴과 char 타입

- 문자 리터럴: 하나의 문자를 작은 따옴표(')로 감싼 것
- 문자 리터럴을 유니코드로 저장할 수 있도록 char 타입 제공

```
char var1 = 'A';    //'A' 문자와 매핑되는 숫자: 65로 대입  
char var3 = '가';   //'가' 문자와 매핑되는 숫자: 44032로 대입
```

각 문자에 해당하는  
코드값이 이미  
정해져있음.(하나로  
통합된 코드값 →  
유니코드)



- char 타입도 정수 타입에 속함

```
char c = 65;        //10진수 65와 매핑되는 문자: 'A'  
char c = 0x0041;    //16진수 0x0041과 매핑되는 문자: 'A'
```

- ch02.sec03.CharExample.java

```
package ch02.sec03;

public class CharExample {
    public static void main(String[] args) {
        char c1 = 'A';           //문자 저장
        char c2 = 65;            //유니코드 직접 저장

        char c3 = '가';         //문자 저장
        char c4 = 44032;        //유니코드 직접 저장

        System.out.println(c1);
        System.out.println(c2);
        System.out.println(c3);
        System.out.println(c4);
    }
}
```

```
char c = '';    //컴파일 에러
char c = ' ';   //공백 하나를 포함해서 초기화
```

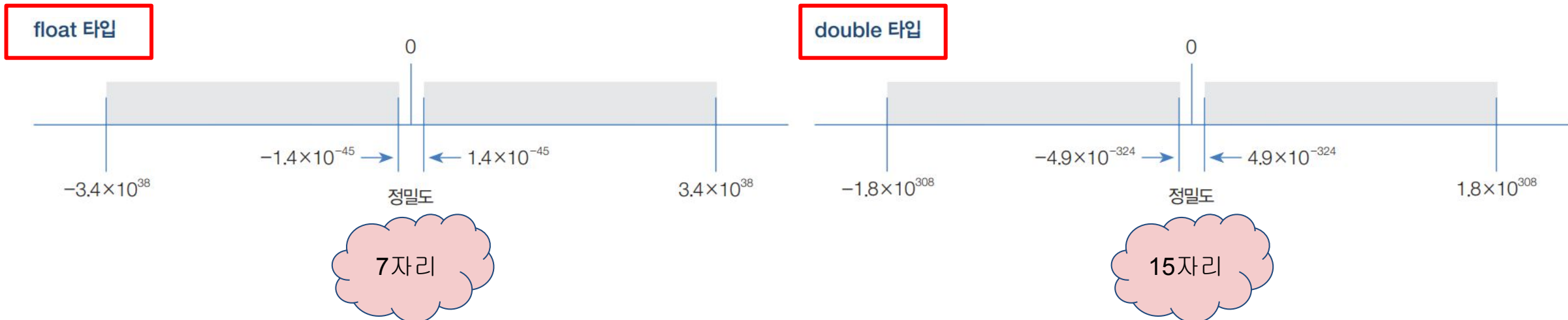
```
A
A
가
가
```

## • float과 double 타입

- 실수 타입에는 float과 double이 있음

타입	메모리 크기		저장되는 값의 허용 범위(양수 기준)	유효 소수 이하 자리
float	4 byte	32 bit	$1.4 \times 10^{-45} \sim 3.4 \times 10^{38}$	7자리
double	8 byte	64 bit	$4.9 \times 10^{-324} \sim 1.8 \times 10^{308}$	15자리

- double 타입이 float 타입보다 큰 실수를 저장할 수 있고 정밀도도 높음



- 실수 리터럴

- 10진수 리터럴

```
double x = 0.25;  
double y = -3.14;
```

- e 또는 E가 포함된 10의 거듭제곱 리터럴

```
double x = 5e2;      //  $5.0 \times 10^2 = 500.0$   
double y = 0.12E-2  //  $0.12 \times 10^{-2} = 0.0012$ 
```

실수 리터럴의 기본 타입은  
**double**

```
double var = 3.14;  
double var = 314e-2;
```

```
float var = 3.14f;  
float var = 3E6F;
```

- ch02.sec04.FloatDoubleExample.java

```
package ch02.sec04;
```

```
public class FloatDoubleExample {  
    public static void main(String[] args) {  
        //정밀도 확인  
        float var1 = 0.1234567890123456789f;  
        double var2 = 0.1234567890123456789;  
        System.out.println("var1: " + var1);  
        System.out.println("var2: " + var2);  
  
        //10의 거듭제곱 리터럴  
        double var3 = 3e6;  
        float var4 = 3e6F;  
        double var5 = 2e-3;  
        System.out.println("var3: " + var3);  
        System.out.println("var4: " + var4);  
        System.out.println("var5: " + var5);  
    }  
}
```

```
var1: 0.12345679  
var2: 0.12345678901234568  
var3: 3000000.0  
var4: 3000000.0  
var5: 0.002
```

double 타입이 float 타입보다 약 2배  
정도의 유효 자릿수를 가진다.

- **boolean** 타입 변수에 대입되는 논리 타입

- 참과 거짓을 의미하는 **true**와 **false**로 구성되며 **boolean** 타입 변수에 대입할 수 있음

```
boolean stop = true;
boolean stop = false;
```

- 주로 두 가지 상태값을 저장하는 경우에 사용. **조건문과 제어문의 실행 흐름을 변경하는 데 사용**

	연산식	
int x = 10;		
boolean result =	(x == 20);	//변수 x의 값이 20인가?
boolean result =	(x != 20);	//변수 x의 값이 20이 아닌가?
boolean result =	(x > 20);	//변수 x의 값이 20보다 큰가?
boolean result =	( 0 < x && x < 20);	//변수 x의 값이 0보다 크고, 20보다 적은가?
boolean result =	( x < 0    x > 200);	//변수 x의 값이 0보다 적거나 200보다 큰가?

조건문과  
반복문의 조건의  
결과는 **boolean**이  
되어야함.  
(switch제외)

- **ch02.sec05.BooleanExample.java**

```
package ch02.sec05;

public class BooleanExample {
    public static void main(String[] args) {
        boolean stop = true;
        if(stop) {
            System.out.println("중지합니다.");
        } else {
            System.out.println("시작합니다.");
        }

        int x = 10;
        boolean result1 = (x == 20); //변수 x의 값이 20인가?
        boolean result2 = (x != 20); //변수 x의 값이 20이 아닌가?
        System.out.println("result1: " + result1);
        System.out.println("result2: " + result2);
    }
}
```

중지합니다.  
result1: false  
result2: true

## • 문자열과 String 타입

- 문자열: 큰따옴표("")로 감싼 문자들
- 문자열을 변수에 저장하려면 **String** 타입을 사용

```
String var1 = "A";
String var2 = "홍길동";
```

```
char var1 = "A";           //컴파일 에러
char var2 = "홍길동";      //컴파일 에러
```

- 이스케이프 문자: 문자열 내부에 역슬래시(\)가 붙은 문자

이스케이프 문자	
\"	" 문자 포함
\'	' 문자 포함
\\	\ 문자 포함
\u16진수	16진수 유니코드에 해당하는 문자 포함
\t	출력 시 탭만큼 띄움
\n	출력 시 줄바꿈(라인피드)
\r	출력 시 캐리지 리턴

escape == 피하다.  
기존의 의미나 사용법을  
피하다.



- **ch02.sec06.StringExample.java**

```
package ch02.sec06;

public class StringExample {
    public static void main(String[] args) {
        String name = "홍길동";
        String job = "프로그래머";
        System.out.println(name);
        System.out.println(job);

        String str = "나는 \"자바\"를 배웁니다.";
        System.out.println(str);

        str = "번호\t이름\t직업 ";
        System.out.println(str);

        System.out.print("나는\n");
        System.out.print("자바를\n");
        System.out.print("배웁니다.");
    }
}
```

```
홍길동
프로그래머
나는 "자바"를 배웁니다.
번호 이름 직업
나는
자바를
배웁니다.
```

## 6 문자열 타입

- 텍스트 블록 문법
  - Java 13부터 지원

```
String str = ""  
...  
"";
```

- ch02.sec06.TextBlockExample.java

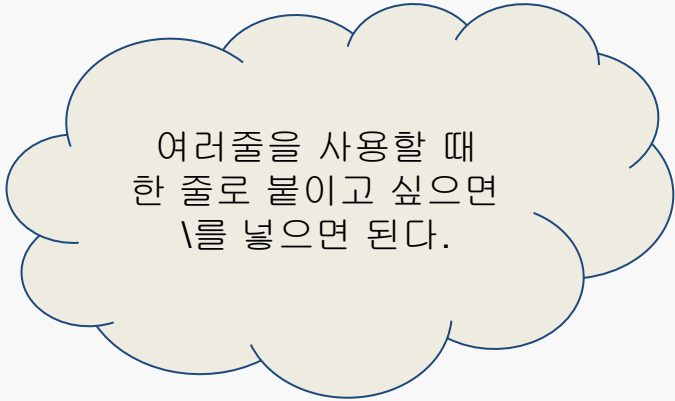
```
package ch02.sec06;
```

```
public class TextBlockExample {  
    public static void main(String[] args) {  
        String str1 = "" +  
            "{\n" +  
            "\\t\"id\": \"winter\", \n" +  
            "\\t\"name\": \"눈송이\" \n" +  
            "}";  
  
        String str2 = ""  
        {  
            "id": "winter",  
            "name": "눈송이"  
        }  
        "";  
  
        System.out.println(str1);  
        System.out.println("-----");  
        System.out.println(str2);  
        System.out.println("-----");  
    }  
}
```


```
{  
    "id": "winter",  
    "name": "눈송이"  
}  
-----  
{  
    "id": "winter",  
    "name": "눈송이"  
}  
-----
```

- ch02.sec06.TextBlockExample.java

```
String str = ""  
나는 자바를 \  
학습합니다.  
나는 자바 고수가 될 겁니다.  
"";  
System.out.println(str);  
}  
}
```



여러줄을 사용할 때  
한 줄로 붙이고 싶으면  
\  
를 넣으면 된다.

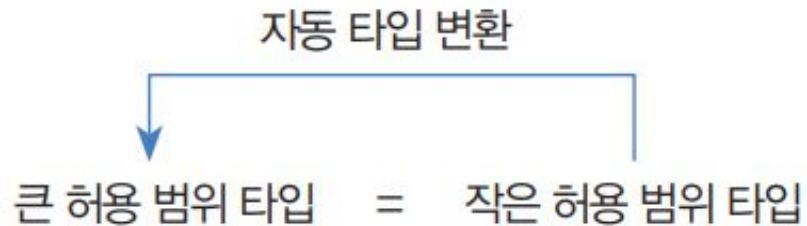


나는 자바를 학습합니다.  
나는 자바 고수가 될 겁니다.

## 7 자동 타입 변환

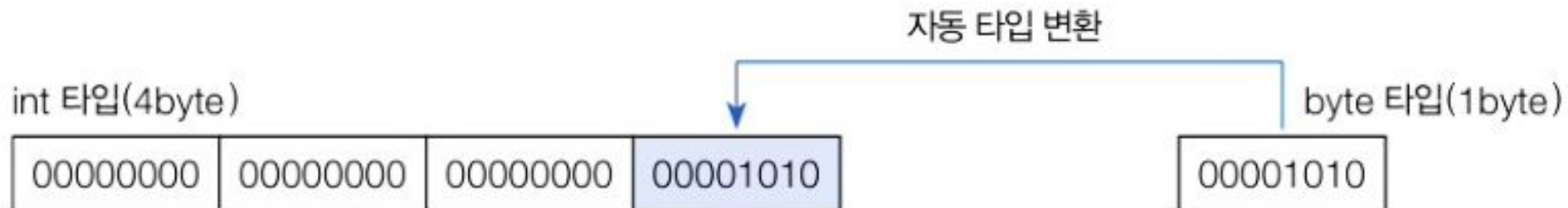
### • 자동 타입 변환

- 데이터 타입을 다른 타입으로 변환하는 것
- 값의 허용 범위가 작은 타입이 허용 범위가 큰 타입으로 대입될 때 발생



byte < short, char < int < long < float < double

```
byte byteValue = 10;
int intValue = byteValue;    //자동 타입 변환됨
```

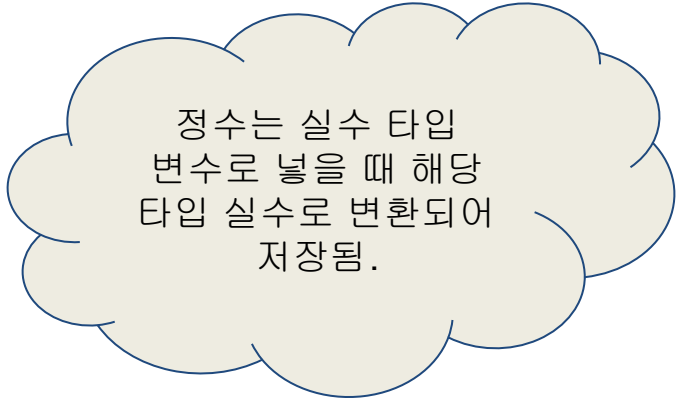


## 7 자동 타입 변환

### • 자동 타입 변환

- 정수 타입이 실수 타입으로 대입  
--> 무조건 자동 타입 변환이 됨

```
long longValue = 50000000000L;  
float floatValue = longValue;    //5.0E9f로 저장됨  
double doubleValue = longValue;  //5.0E9로 저장됨
```



정수는 실수 타입  
변수로 넣을 때 해당  
타입 실수로 변환되어  
저장됨.

- char 타입이 int 타입으로 변환될 때 유니 코드 값이 대입

```
char charValue = 'A';  
int intValue = charValue;    //65가 저장됨
```

## □ ch02.sec07.PromotionExample.java

```

public class PromotionExample {
    public static void main(String[] args) {
        //자동 타입 변환
        byte byteValue = 10;
        int intValue = byteValue;
        System.out.println("intValue: " + intValue);

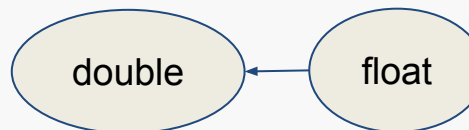
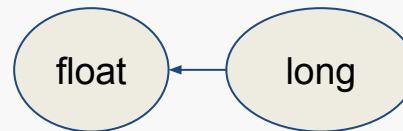
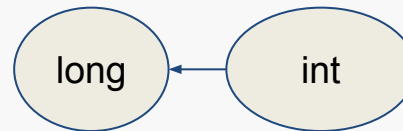
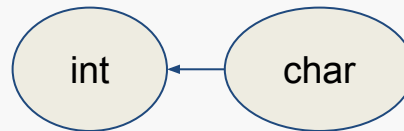
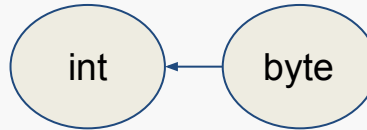
        char charValue = '가';
        intValue = charValue;
        System.out.println("가의 유니코드: " + intValue);

        intValue = 50;
        long longValue = intValue;
        System.out.println("longValue: " + longValue);

        longValue = 100;
        float floatValue = longValue;
        System.out.println("floatValue: " + floatValue);

        floatValue = 100.5F;
        double doubleValue = floatValue;
        System.out.println("doubleValue: " + doubleValue);
    }
}

```



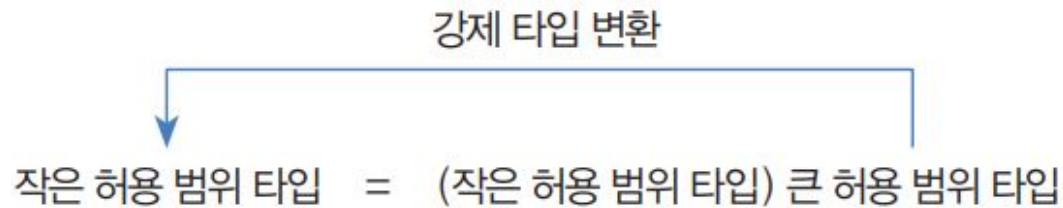
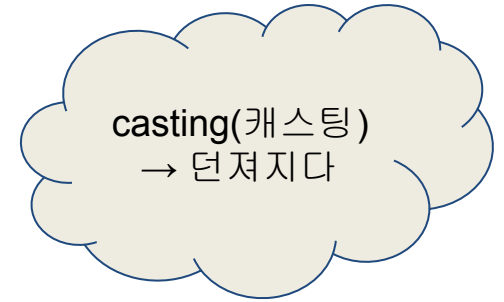
```

intValue: 10
가의 유니코드: 44032
longValue: 50
floatValue: 100.0
doubleValue: 100.5

```

## • 캐스팅 연산자로 강제 타입 변환하기

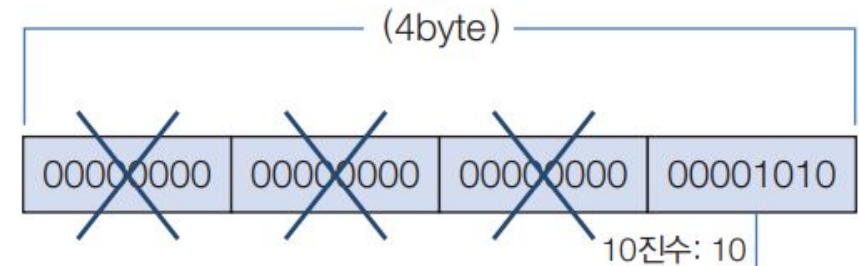
- 큰 허용 범위 타입을 작은 허용 범위 타입으로 쪼개어서 저장하는 것
- 캐스팅 연산자로 **괄호()**를 사용하며, 괄호 안에 들어가는 타입은 쪼개는 단위



- int → byte

```
int intValue = 10;
byte byteValue = (byte) intValue;
```

```
int intValue = 10;
byte byteValue = (byte) intValue;
```



원래 값이 보존됨



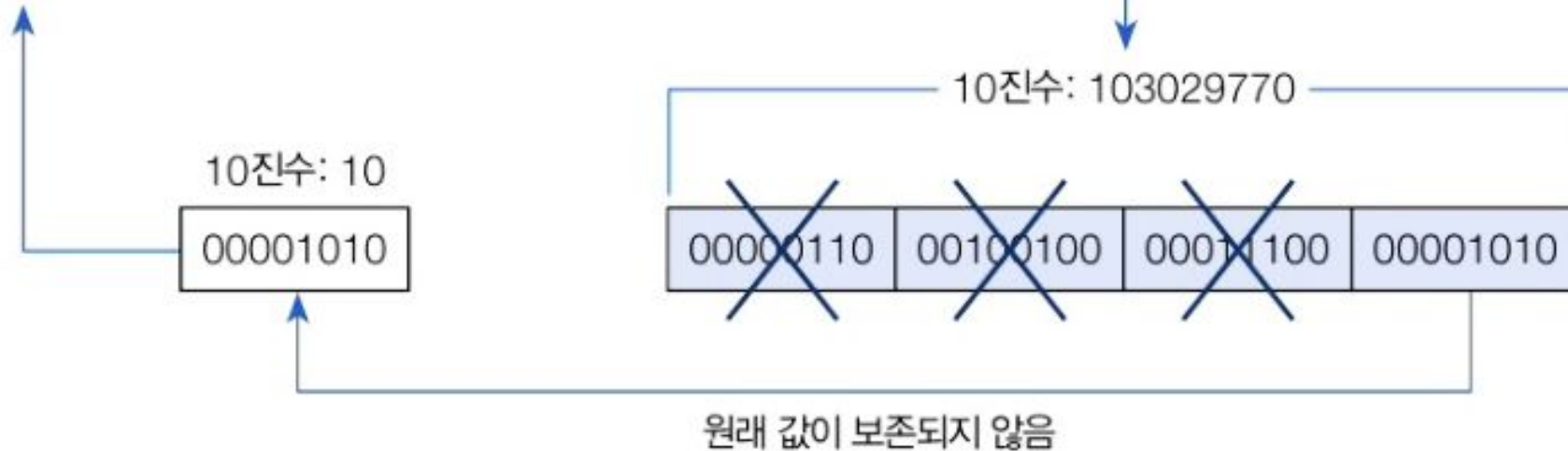
- 캐스팅 연산자로 강제 타입 변환하기

- int → byte

- 원래 값이 보존되지 않는 경우

타입변환 불가능

```
int intValue = 103029770;  
byte byteValue = (byte) intValue;
```



- 캐스팅 연산자로 강제 타입 변환하기

- long → int

```
long longValue = 300;  
int intValue = (int) longValue;    //강제 타입 변환 후에 300이 그대로 유지
```

- int → char

```
int intValue = 65;  
char charValue = (char) intValue;  
System.out.println(charValue);    //'A'가 출력
```

- 실수 → 정수

```
double doubleValue = 3.14;  
int intValue = (int) doubleValue;    //intValue는 정수 부분인 3만 저장
```

- ch02.sec08.CastingExample.java

```
package ch02.sec08;

public class CastingExample {
    public static void main(String[] args) {
        int var1 = 10;
        byte var2 = (byte) var1;
        System.out.println(var2);    //강제 타입 변환 후에 10이 그대로 유지

        long var3 = 300;
        int var4 = (int) var3;
        System.out.println(var4);    //강제 타입 변환 후에 300이 그대로 유지

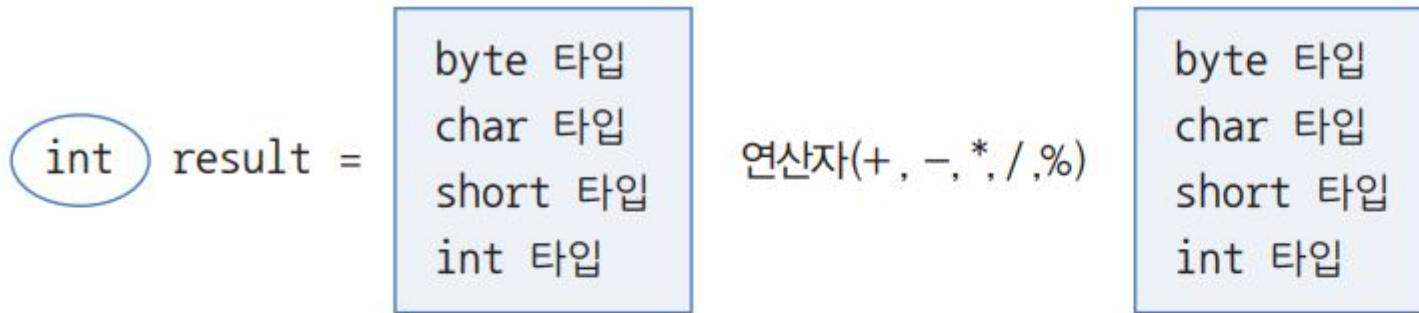
        int var5 = 65;
        char var6 = (char) var5;
        System.out.println(var6);    //'A'가 출력

        double var7 = 3.14;
        int var8 = (int) var7;
        System.out.println(var8);    //3이 출력
    }
}
```

```
10
300
A
3
```

## • 연산식에서 int 타입의 자동 변환

- 정수 타입 변수가 산술 연산식에서 피연산자로 사용되면  
→ **int** 타입보다 작은 **byte**, **short** 타입 변수는 **int** 타입으로 자동 변환되어 연산 수행

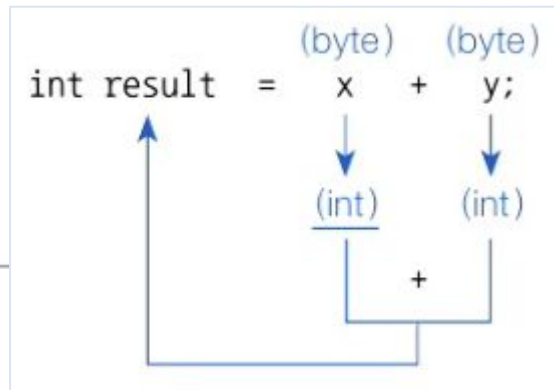


### byte 타입 변수가 피연산자로 사용된 경우

```
byte x = 10;
byte y = 20;
byte result = x + y; //컴파일 에러
int result = x + y;
```

### int 타입 변수가 피연산자로 사용된 경우

```
int x = 10;
int y = 20;
int result = x + y;
```



## 연산식에서 자동 타입 변환

### • 연산식에서 long 타입의 자동 변환

- 다른 타입의 피연산자들이 long 타입으로 변환되어 계산



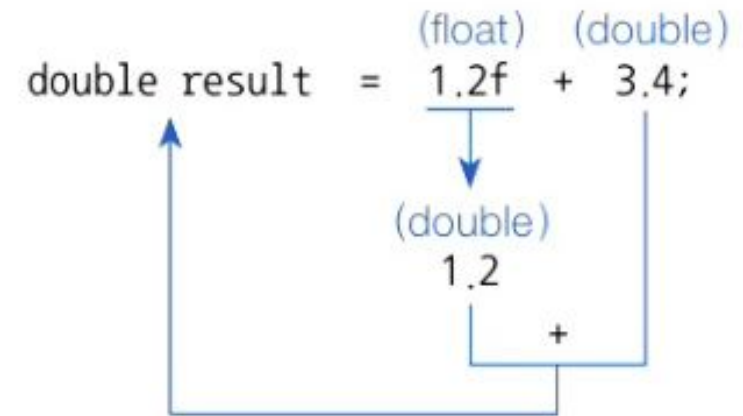
연산시 피연산자의 타입이 하나라도 double이면 double로 변환되어 연산함. → 결과는 무조건 실수

### • 연산식에서 실수 타입의 자동 변환

- float 만 있는 경우 float로 계산

```
float result = 1.2f + 3.4f; //컴파일: float result = 4.6f;
```

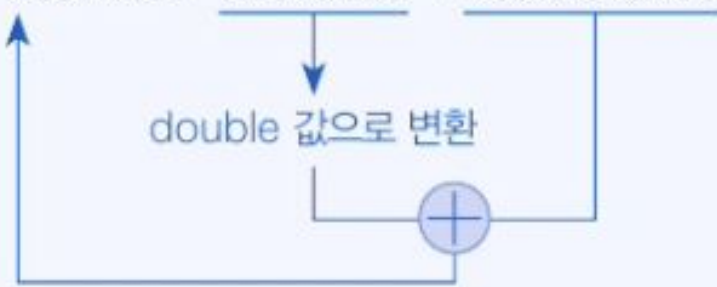
- double이 있는 경우 모두 double로 자동 변환 후 계산



- 연산식에서 실수 타입의 자동 변환

- int 타입과 double 타입이 연산에 있는 경우 → double 타입으로 자동변환

```
int intValue = 10;
double doubleValue = 5.5;
double result = intValue + doubleValue;    //10.0 + 5.5
```



연산시 피연산자의  
타입이 하나라도  
double이면  
double로 변환되어  
연산함. → 결과는  
무조건 실수

- 정수 타입으로 계산을 원하는 경우 → 강제 형변환

```
int intValue = 10;
double doubleValue = 5.5;
int result = intValue + (int) doubleValue;    //10 + 5
```

- 연산식에서 실수 타입의 자동 변환

```
int x = 1;
int y = 2;
double result = x / y;
System.out.println(result);    //0.5가 출력될까요?
```



→ 결과는 0.0

- 실수 연산을 원한다면 피연산자 중 하나를 **double**로 강제 형변환해야 함

방법 1	방법 2	방법 3
<pre>int x = 1; int y = 2; double result = (double) x / y; System.out.println(result);</pre>	<pre>int x = 1; int y = 2; double result = x / (double) y; System.out.println(result);</pre>	<pre>int x = 1; int y = 2; double result = (double) x / (double) y; System.out.println(result);</pre>

- ch02.sec09.OperationPromotionExample.java

```
public class OperationPromotionExample {  
    public static void main(String[] args) {  
        byte result1 = 10 + 20; //컴파일 단계에서 연산  
        System.out.println("result1: " + result1);  
  
        byte v1 = 10;  
        byte v2 = 20;  
        int result2 = v1 + v2; //int 타입으로 변환 후 연산  
        System.out.println("result2: " + result2);  
  
        byte v3 = 10;  
        int v4 = 100;  
        long v5 = 1000L;  
        long result3 = v3 + v4 + v5; //long 타입으로 변환 후 연산  
        System.out.println("result3: " + result3);  
  
        char v6 = 'A';  
        char v7 = 1;  
        int result4 = v6 + v7; //int 타입으로 변환 후 연산  
        System.out.println("result4: " + result4);  
        System.out.println("result4: " + (char)result4);  
    }  
}
```

```
result1: 30  
result2: 30  
result3: 1110  
result4: 66  
result4: B
```



- ch02.sec09.OperationPromotionExample.java

```
int v8 = 10;  
int result5 = v8 / 4; //정수 연산의 결과는 정수  
System.out.println("result5: " + result5);
```

```
int v9 = 10;  
double result6 = v9 / 4.0; //double 타입으로 변환 후 연산  
System.out.println("result6: " + result6);
```

```
int v10 = 1;  
int v11 = 2;  
double result7 = (double) v10 / v11; //double 타입으로 변환 후 연산  
System.out.println("result7: " + result7);
```

```
}
```

```
}
```

```
result5: 2  
result6: 2.5  
result7: 0.5
```

- String 타입 자동 변환

- 정수 + 문자열 또는 문자열 + 정수 → 문자열

```
int value = 3 + 7;    → int value = 10;
String str = "3" + 7; → String str = "3" + "7"; → String str = "37";
String str = 3 + "7"; → String str = "3" + "7"; → String str = "37";
```

- 주의사항: 정수 + 정수 + 문자열 → 정수 합계 계산 후 문자열로 결합

```
int value = 1 + 2 + 3;    → int value = 3 + 3;    → int value = 6;
String str = 1 + 2 + "3"; → String str = 3 + "3"; → String str = "33";
String str = 1 + "2" + 3; → String str = "12" + 3; → String str = "123";
String str = "1" + 2 + 3; → String str = "12" + 3; → String str = "123";
```

```
String str = "1" + (2 + 3); → String str = "1" + 5; → String str = "15";
```

+가 결합연산자인  
경우, 하나라도  
String이면 결과는  
String이 된다.



- ch02.sec09.StringConcatExample.java

```
package ch02.sec09;

public class StringConcatExample {
    public static void main(String[] args) {
        //숫자 연산
        int result1 = 10 + 2 + 8;
        System.out.println("result1: " + result1);

        //결합 연산
        String result2 = 10 + 2 + "8";
        System.out.println("result2: " + result2);

        String result3 = 10 + "2" + 8;
        System.out.println("result3: " + result3);

        String result4 = "10" + 2 + 8;
        System.out.println("result4: " + result4);

        String result5 = "10" + (2 + 8);
        System.out.println("result5: " + result5);
    }
}
```



```
result1: 20
result2: 128
result3: 1028
result4: 1028
result5: 1010
```

## • String 타입 변환하기

변환 타입	사용 예
String → byte	String str = "10"; <b>byte value = Byte.parseByte(str);</b>
String → short	String str = "200"; <b>short value = Short.parseShort(str);</b>
String → int	String str = "300000"; <b>int value = Integer.parseInt(str);</b>
String → long	String str = "400000000000"; <b>long value = Long.parseLong(str);</b>
String → float	String str = "12.345"; <b>float value = Float.parseFloat(str);</b>
String → double	String str = "12.345"; <b>double value = Double.parseDouble(str);</b>
String → boolean	String str = "true"; <b>boolean value = Boolean.parseBoolean(str);</b>

String 변수에 들어있는 값을 다른 타입의 숫자로 변환이 가능한 경우에만 사용 가능!

String n = "100";  
html의 input태그에서 입력한 value속성값은 기본 text인 경우 문자열로 인식

- ch02.sec10.PrimitiveAndStringConversionExample.java

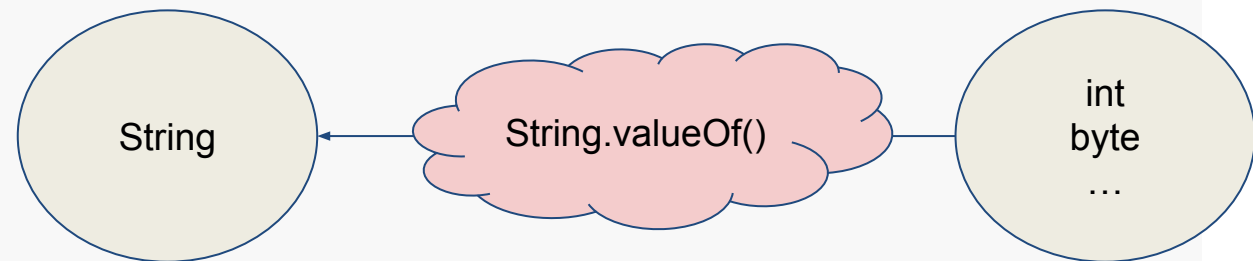
```
package ch02.sec10;

public class PrimitiveAndStringConversionExample {
    public static void main(String[] args) {
        int value1 = Integer.parseInt("10");
        double value2 = Double.parseDouble("3.14");
        boolean value3 = Boolean.parseBoolean("true");

        System.out.println("value1: " + value1);
        System.out.println("value2: " + value2);
        System.out.println("value3: " + value3);

        String str1 = String.valueOf(10);
        String str2 = String.valueOf(3.14);
        String str3 = String.valueOf(true);

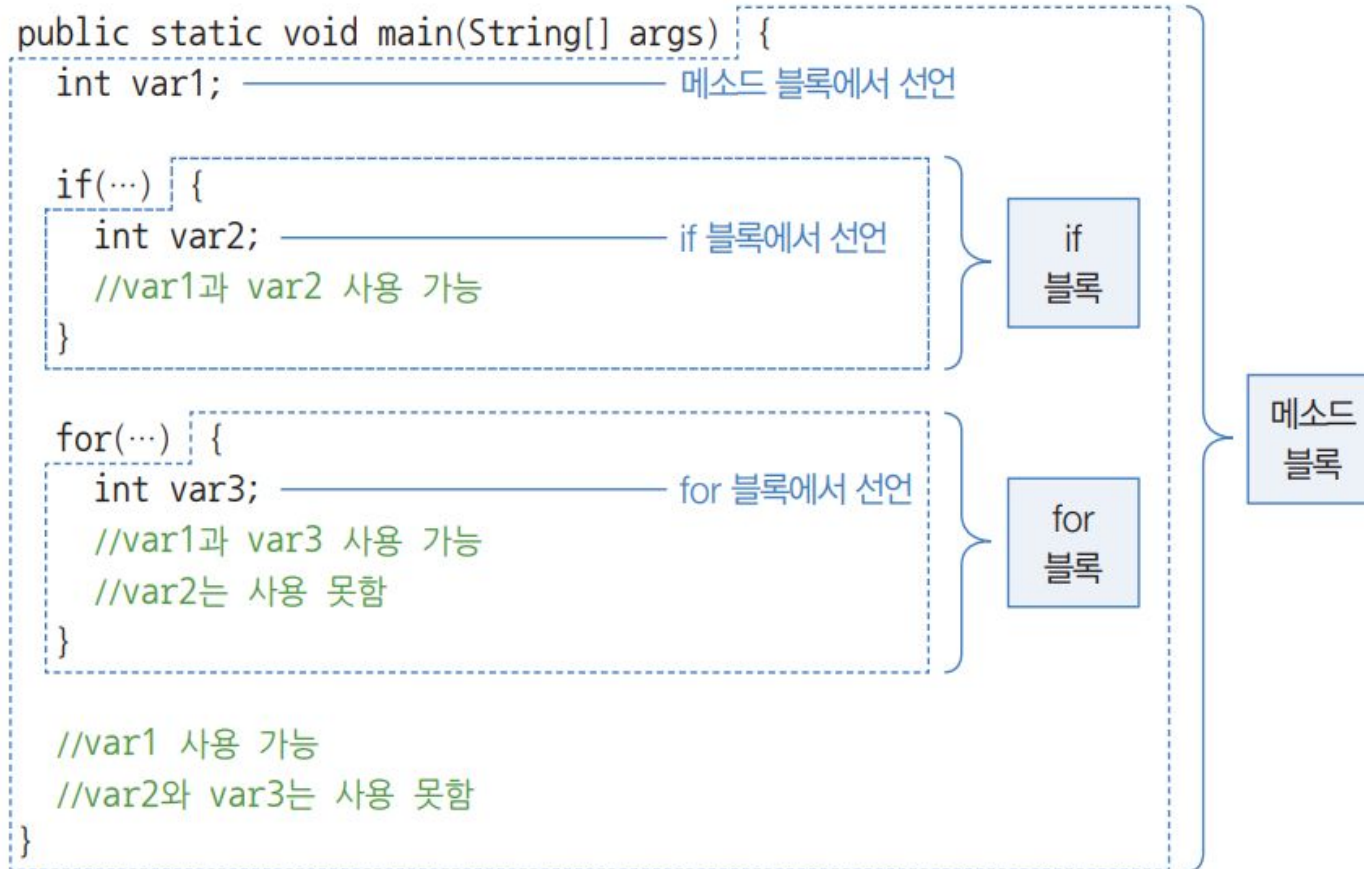
        System.out.println("str1: " + str1);
        System.out.println("str2: " + str2);
        System.out.println("str3: " + str3);
    }
}
```



```
value1: 10
value2: 3.14
value3: true
str1: 10
str2: 3.14
str3: true
```

## • 변수 범위를 나타내는 중괄호 {} 블록

- 조건문과 반복문의 중괄호 {} 블록 내에 선언된 변수는 해당 중괄호 {} 블록 내에서만 사용 가능



지역변수 → 블록내에서 선언된 변수는 **해당 블록내에서만 사용 가능**

- **var1**은 **main()**내, **if, for**문 내에서 사용 가능
- **var2**는 **if**문 내에서만 사용 가능
- **var3**는 **for**문 내에서만 사용 가능

- **ch02.sec11.VariableScopeExample.java**

```
package ch02.sec11;
```

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

```
        int v1 = 15;
```

```
        if(v1>10) {
```

```
            int v2 = v1 - 10;
```

```
        }
```

```
        int v3 = v1 + v2 + 5; //v2 변수를 사용할 수 없기 때문에 컴파일 에러 발생
```

```
    }
```

```
}
```

- **println() 메소드로 변수값 출력하기**
  - 모니터에 값을 출력하기 위해 `System.out.println()` 이용

`System.` + `out.` + `println(리터럴 또는 변수);`

↓                      ↓                      ↓

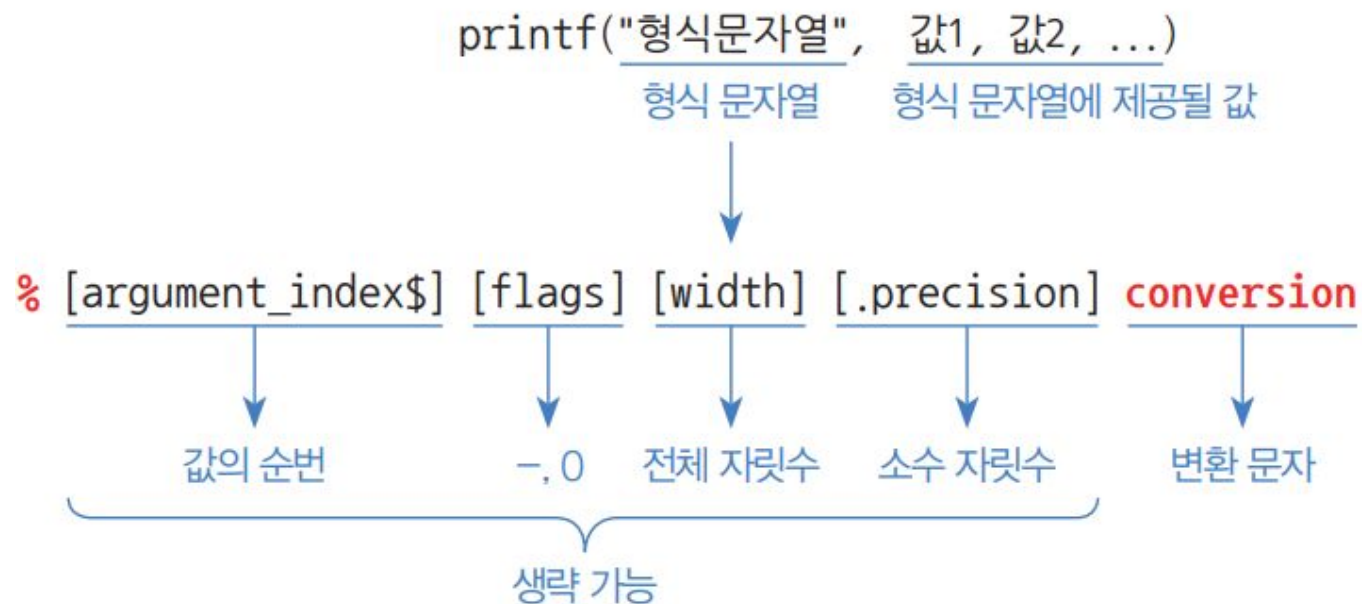
시스템으로      출력하는데      괄호 안의 내용을 출력하고 행을 바꿔라

- 출력 방법에 따라 `println()` 이외에도 다음과 같이 `print()`, `printf()`를 사용할 수 있음

메소드	의미
<code>println(내용);</code>	괄호 안의 내용을 출력하고 행을 바꿔라.
<code>print(내용);</code>	괄호 안의 내용을 출력하고 행은 바꾸지 말아라.
<code>printf("형식문자열", 값1, 값2, ...);</code>	형식 문자열에 맞추어 뒤의 값을 출력해라.



- **println() 메소드로 변수값 출력하기**
  - **printf()의 형식 문자열**
    - %와 **conversion**(변환 문자)를 필수로 작성하고 나머지는 생략 가능



- **println()** 메소드로 변수값 출력하기
  - **printf()**의 형식 문자열

```
System.out.printf("이름: %s", "감자바"); → 이름: 감자바
System.out.printf("나이: %d", 25 ); → 나이: 25
```

```
System.out.printf("이름: %1$s, 나이: %2$d", "김자바", 25); → 이름: 김자바, 나이: 25
```

%1, %2는 순서에  
해당하는 번호

## • println() 메소드로 변수값 출력하기

### ○ printf()의 형식 문자열

형식화된 문자열		설명	출력 형태
정수	%d	정수	123
	%6d	6자리 정수. 왼쪽 빈자리 공백	___123
	%-6d	6자리 정수. 오른쪽 빈자리 공백	123___
	%06d	6자리 정수. 왼쪽 빈자리 0 채움	000123
실수	%10.2f	정수 7자리+소수점+소수 2자리. 왼쪽 빈자리 공백	____123.45
	%-10.2f	정수 7자리+소수점+소수 2자리. 오른쪽 빈자리 공백	123.45____
	%010.2f	정수 7자리+소수점+소수 2자리. 왼쪽 빈자리 0 채움	0000123.45
문자열	%s	문자열	abc
	%6s	6자리 문자열. 왼쪽 빈자리 공백	___abc
	%-6s	6자리 문자열. 오른쪽 빈자리 공백	abc___
특수 문자	\t	탭(tab)	
	\n	줄바꿈	
	%%	%	%

- ch02.sec12.PrintfExample.java

```
package ch02.sec12;

public class PrintfExample {
    public static void main(String[] args) {
        int value = 123;
        System.out.printf("상품의 가격:%d원\n", value);
        System.out.printf("상품의 가격:%6d원\n", value);
        System.out.printf("상품의 가격:%-6d원\n", value);
        System.out.printf("상품의 가격:%06d원\n", value);

        double area = 3.14159 * 10 * 10;
        System.out.printf("반지름이 %d인 원의 넓이:%10.2f\n", 10, area);

        String name = "홍길동";
        String job = "도적";
        System.out.printf("%6d | %-10s | %10s\n", 1, name, job);
    }
}
```

```
상품의 가격:123원
상품의 가격: 123원
상품의 가격:123 원
상품의 가격:000123원
반지름이 10인 원의 넓이: 314.16
  1 | 홍길동      |      도적
```

## • Scanner 타입 변수 활용하기

- Scanner 타입 변수를 선언
- 대입 연산자 =를 사용해서 new 연산자로 생성한 Scanner 객체를 변수에 대입

생성된 Scanner를 변수에 대입

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

scanner 변수 선언      Scanner 객체 생성

- scanner.nextLine()을 실행하면 키보드로 입력된 내용을 문자열로 읽고 좌측 String 변수에 저장

읽은 문자열을 String 변수에 저장

```
String inputData = scanner.nextLine();
```

String 변수 선언      [Enter] 키를 누르면 입력된 문자열을 읽음

키보드로 입력된 내용을  
변수에 저장  
→ Scanner의  
nextLine()을 이용한  
입력은 문자열로 취급

- ch02.sec13.ScannerExample.java

```
package ch02.sec13;
```

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

```
public class ScannerExample {  
    public static void main(String[] args) throws Exception {  
        Scanner scanner = new Scanner(System.in);
```

```
        System.out.print("x 값 입력: ");  
        String strX = scanner.nextLine();  
        int x = Integer.parseInt(strX);
```

```
        System.out.print("y 값 입력: ");  
        String strY = scanner.nextLine();  
        int y = Integer.parseInt(strY);
```

```
        int result = x + y;  
        System.out.println("x + y: " + result);  
        System.out.println();
```

ch02.sec13 이외의 패키지에  
있는 클래스는 import로  
사용할 클래스의 위치를  
지정해주어야 한다.

```
x 값 입력: 3  
y 값 입력: 5  
x + y: 8
```

- ch02.sec13.ScannerExample.java

```
while(true) {
    System.out.print("입력 문자열: ");
    String data = scanner.nextLine();
    if(data.equals("q")) {
        break;
    }
    System.out.println("출력 문자열: " + data);
    System.out.println();
}
```

```
System.out.println("종료");
```

```
}
```

입력 문자열: Hello

출력 문자열: Hello

입력 문자열: 안녕하세요

출력 문자열: 안녕하세요

입력 문자열: q

종료

```
boolean result = data . equals("문자열");
```

