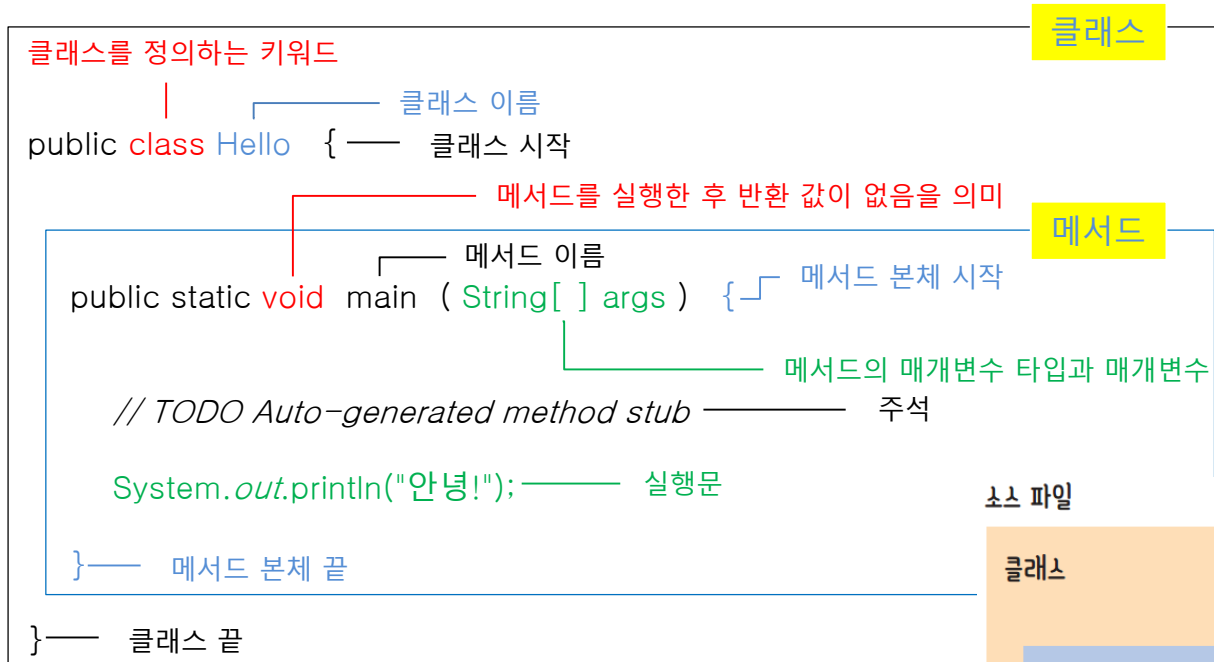


02. 자바 프로그램의 구조와 기본 문법



자바 프로그램 기본 구조

■ Hello 프로그램 구조



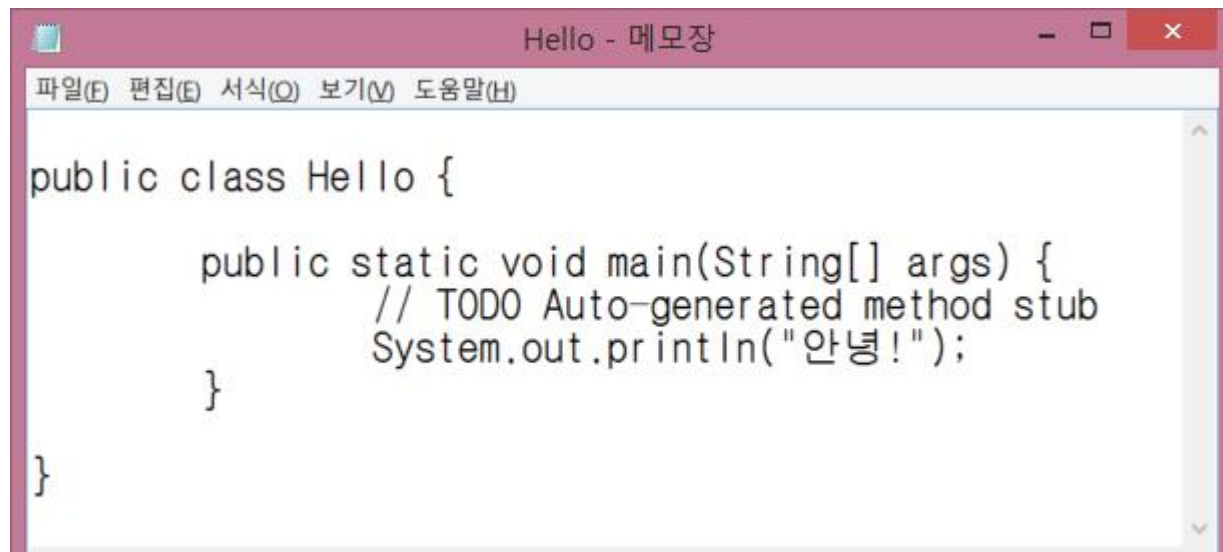
자바 프로그램 기본 구조

■ Hello 프로그램 구조

- 클래스 : 객체 지향 언어에서 프로그램을 개발하는 단위
- 메서드 : 수행할 작업을 나열한 코드의 모임
- 실행문 : 작업을 지시하는 변수 선언, 값 저장, 메서드 호출 등의 코드
- 주석문
 - 행 주석 : //
 - 범위 주석 : /* */
 - 문서 주석 : /** */

■ Hello 프로그램의 확장

- [sec01/Hello](#)



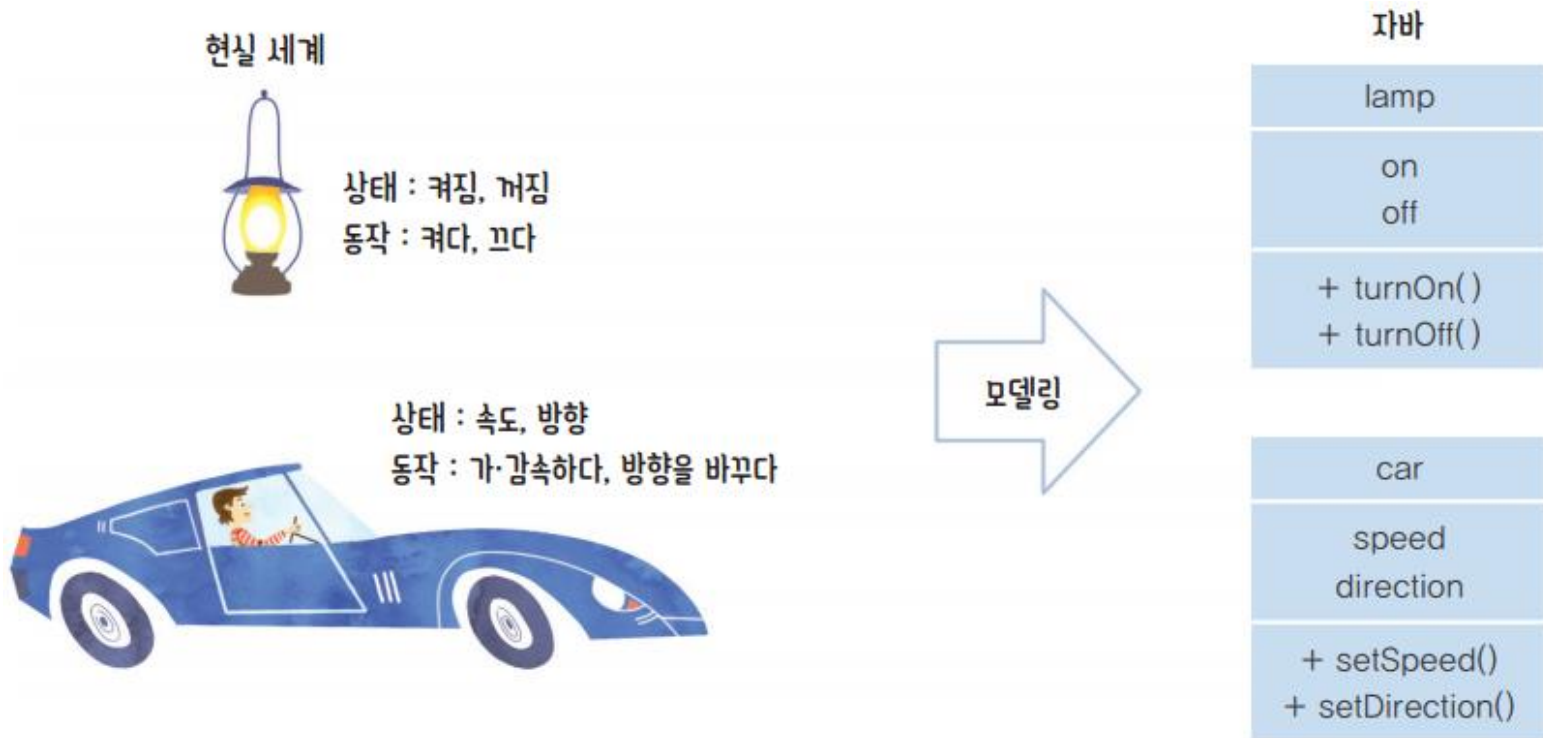
```
public class Hello {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        System.out.println("안녕!");  
    }  
  
}
```



객체지향 기초

■ 객체의 개념

- 소프트웨어 객체는 현실 세계의 객체를 필드와 메서드로 모델링한 것
- 소프트웨어 객체는 상태를 필드(Field)로 정의하고, 동작을 메서드(Method)로 정의.
- 필드는 객체 내부에 선언된 변수를 의미하고, 메서드는 객체 내부에 정의된 동작



객체지향 기초

■ 절차 지향 프로그래밍

- 일련의 동작을 순서에 맞추어 단계적으로 실행하도록 명령어를 나열
- 데이터를 정의하는 방법보다는 명령어의 순서와 흐름에 중점
- 수행할 작업을 예상할 수 있어 직관적인데, 규모가 작을 때는 프로그래밍과 이해하기가 용이
- 소프트웨어는 계산 위주이므로 절차 지향 프로그래밍이 적합

■ 객체 지향 프로그래밍

- 소프트웨어의 규모가 커지면서 동작과 분리되어 전 과정에서 서로 복잡하게 얽혀 있는 데이터를 사용했기 때문에 절차 지향 프로그래밍 방식의 한계
- 절차 지향 프로그램은 추후 변경하거나 확장하기도 어려움
- 현실 세계를 객체 단위로 프로그래밍하며, 객체는 필드(데이터)와 메서드(코드)를 하나로 묶어 표현



객체지향 기초

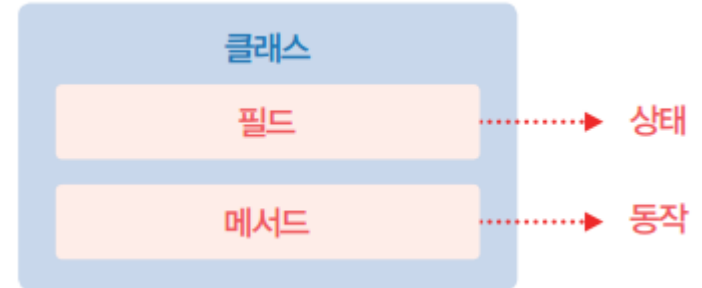
■ 객체와 클래스



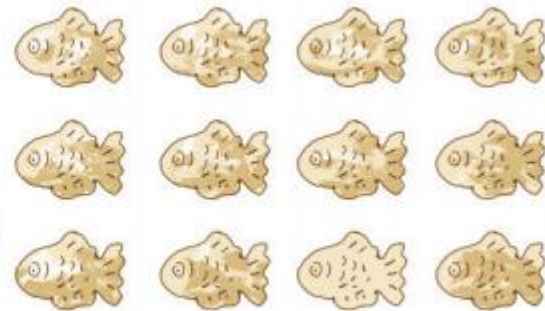
형틀 = 클래스



제품 = 객체



인스턴스화



붕어빵
한 마리가
한 마리가
객체에 해당

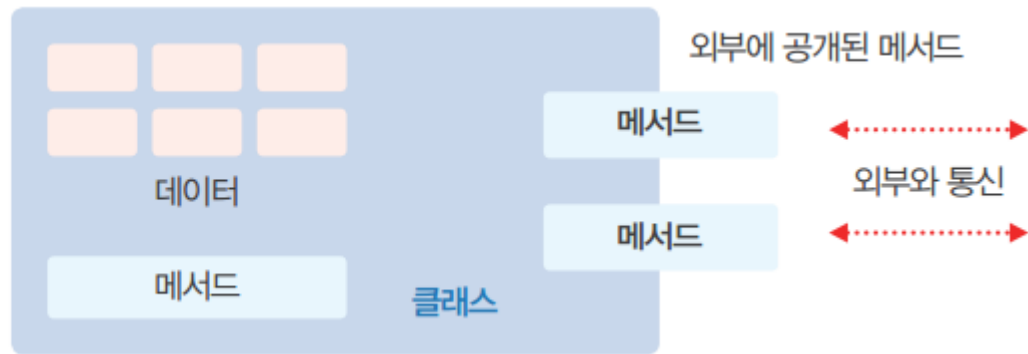
인스턴스 = 붕어빵의 실제



객체 지향 프로그래밍

■ 특징

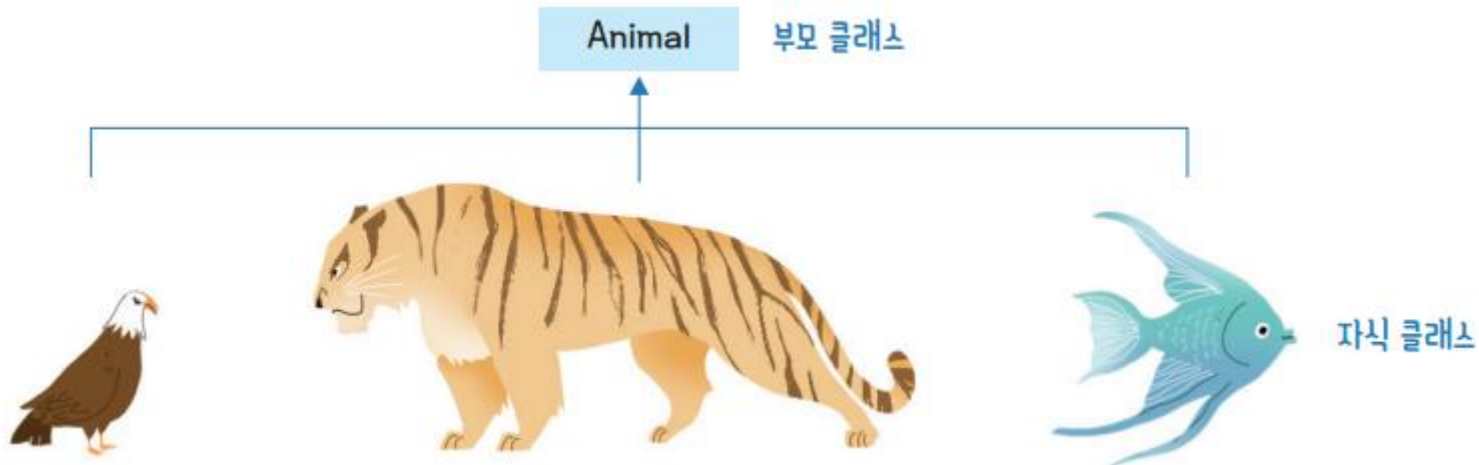
- **캡슐화**(정보 은닉) : 관련된 필드와 메서드를 하나의 캡슐처럼 포장해 세부 내용을 외부에서 알 수 없도록 감추는 것



객체 지향 프로그래밍

■ 특징

- **상속** : 자녀가 부모 재산을 상속받아 사용하듯이 상위 객체를 상속받은 하위 객체가 상위 객체의 메서드와 필드를 사용하는 것
- 상속은 개발된 객체를 재사용하는 방법 중 하나



식별자

■ 규칙

- 문자, 언더바(_), \$로 시작해야 한다. 한글도 가능하며, 영문자는 대·소문자를 구분한다.
- +, - 등 연산자를 포함하면 안 된다.
- 자바 키워드를 사용하면 안 된다.
- 길이에 제한이 없다.

잘못된 식별자 : %5, a+b, 1b
올바른 식별자 : radius, \$a, _int

■ 자바 키워드

분류	키워드
데이터 타입	byte, char, short, int, long, float, double, boolean
접근 지정자	private, protected, public
제어문	if, else, for, while, do, break, continue, switch, case
클래스와 객체	class, interface, enum, extends, implements, new, this, super, instanceof, null
예외 처리	try, catch, finally, throw, throws
기타	abstract, assert, const, default, false, final, import, native, package, return, static, strictfp, synchronized, transient, true, void, volatile



식별자

■ 관례

- 변수와 메서드는 모두 소문자로 표기. 단, 복합 단어일 때는 두 번째 단어부터 단어의 첫 자만 대문자로 표기
- 클래스와 인터페이스는 첫 자만 대문자로 표기 하고 나머지는 소문자로 표기. 단, 복합 단어일 때는 두 번째 단어부터 단어의 첫 자만 대문자로 표기
- 상수는 전체를 대문자로 표기. 단, 복합 단어일 때는 단어를 언더바(_)로 연결

```
int thisYear;  
String currentPosition;  
boolean isEmpty;  
public int getYear( ) { }
```

```
public class HelloDemo { }  
public interface MyRunnable { }
```

```
final int NUMBER_ONE = 1;  
final double PI = 3.141592;
```



데이터 타입

■ 의미

- 값과 값을 다룰 수 있는 연산의 집합을 의미

■ 종류



데이터 타입

■ 기억 공간 크기 및 기본 값

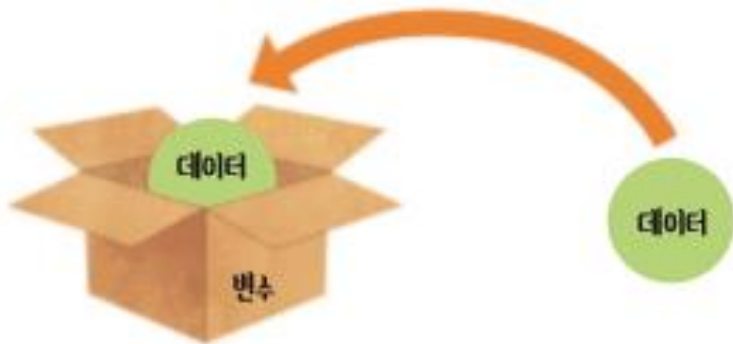
분류	기초 타입	기억 공간 크기	기본 값	값의 범위
정수	byte	8비트	0	-128 ~ 127
	short	16비트	0	-32,768 ~ 32,767
	int	32비트	0	-2,147,483,648 ~ 2,147,483,647
	long	64비트	0L	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
문자	char	16비트	null	0('����') ~ 65,535('�FFFF')
실수	float	32비트	0.0f	약 $\pm 3.4 \times 10^{-38} \sim \pm 3.4 \times 10^{+38}$
	double	64비트	0.0d	약 $\pm 1.7 \times 10^{-308} \sim \pm 1.7 \times 10^{+308}$
논리	boolean	8비트	false	true와 false



변수

■ 의미

- 프로그램은 기억 공간에 데이터를 보관하고, 각 기억 공간을 변수Variable로 구분
- 변수는 데이터를 담는 상자와 같은 것으로 종류가 다양한데, 이를 구분하려고 데이터 타입을 사용



변수

■ 리터럴

- 프로그램 내부에서 값을 정의해 변수를 초기화할 수 있는데, 그 값을 리터럴

■ 정수

```
int fifteen = 15;           // 10진수
byte fifteen = 0b1111;      // 2진수 15
short fifteen = 017;        // 8진수 15
int fifteen = 0xF;          // 16진수 15
long lightSpeed = 300000L;  // L로 long 타입임을 명시
```

■ 실수

```
double half = 0.5;          // 일반 표기법
double half = 5E-1;         // 지수 표기법으로  $5 \times 10^{-1}$ 을 의미
float pi = 3.14159;         // 오류
float pi = 3.14159F;        // F는 float 타입임을 명시
double pi = 3.14159;
```

가수이다.

지수이다.

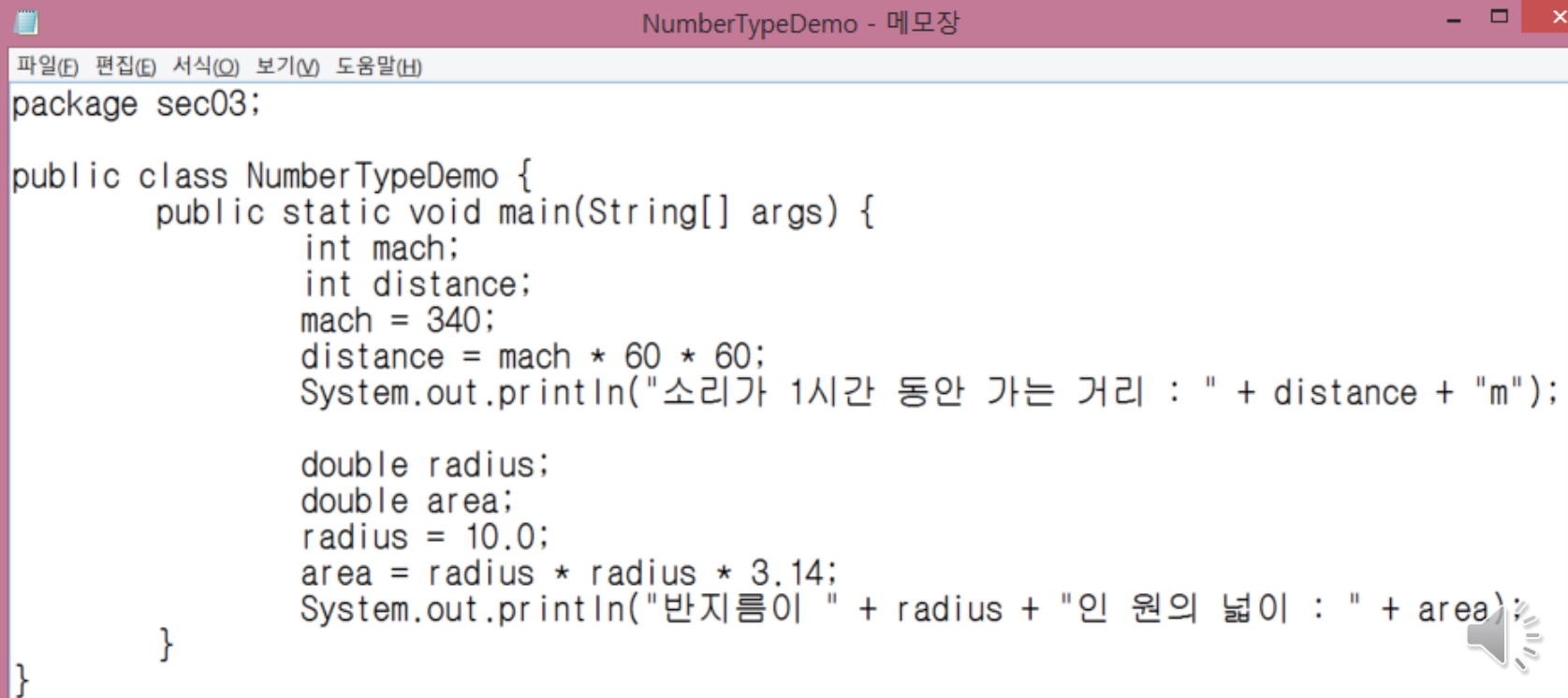


변수

■ 예제

- 코드 : [sec03/NumberTypeDemo](#)
- 실행 결과

소리가 1시간 동안 가는 거리 : 1224000m
반지름이 10.0인 원의 넓이 : 314.0



```
package sec03;

public class NumberTypeDemo {
    public static void main(String[] args) {
        int mach;
        int distance;
        mach = 340;
        distance = mach * 60 * 60;
        System.out.println("소리가 1시간 동안 가는 거리 : " + distance + "m");

        double radius;
        double area;
        radius = 10.0;
        area = radius * radius * 3.14;
        System.out.println("반지름이 " + radius + "인 원의 넓이 : " + area);
    }
}
```

변수

■ 문자

```
char c = 'A';           // 문자
char c = 65;            // 일종의 정수 타입이기 때문에 65 대입 가능
char c = '\u0041';      // 유니코드 값으로 대입
char c = "A";           // "A"는 문자가 아니라 문자열이므로 오류
```

■ 논리

```
boolean condition = true; // 논리 리터럴 true와 false 중 하나
```

■ 예제

- 코드 [sec03/CharBoolDemo](#)
- 실행 결과

```
가
가
true가 아니면 false입니다.
```

```
public class CharBoolDemo {
    public static void main(String[] args) {
        char ga1 = '가';
        char ga2 = 'Wuac00';

        boolean cham = true;
        boolean geojit = false;

        System.out.println(ga1);
        System.out.println((int)ga1);
        System.out.println(ga2);
        System.out.println(++ga2);
        System.out.println(cham + "가 아니면 " + geojit + "입니다.");
    }
}
```


변수

■ 변수 사용

```
int weight;           // 정수 타입의 weight 변수 선언  
double x, y, z;       // 3개의 변수를 ,로 연결해 선언
```



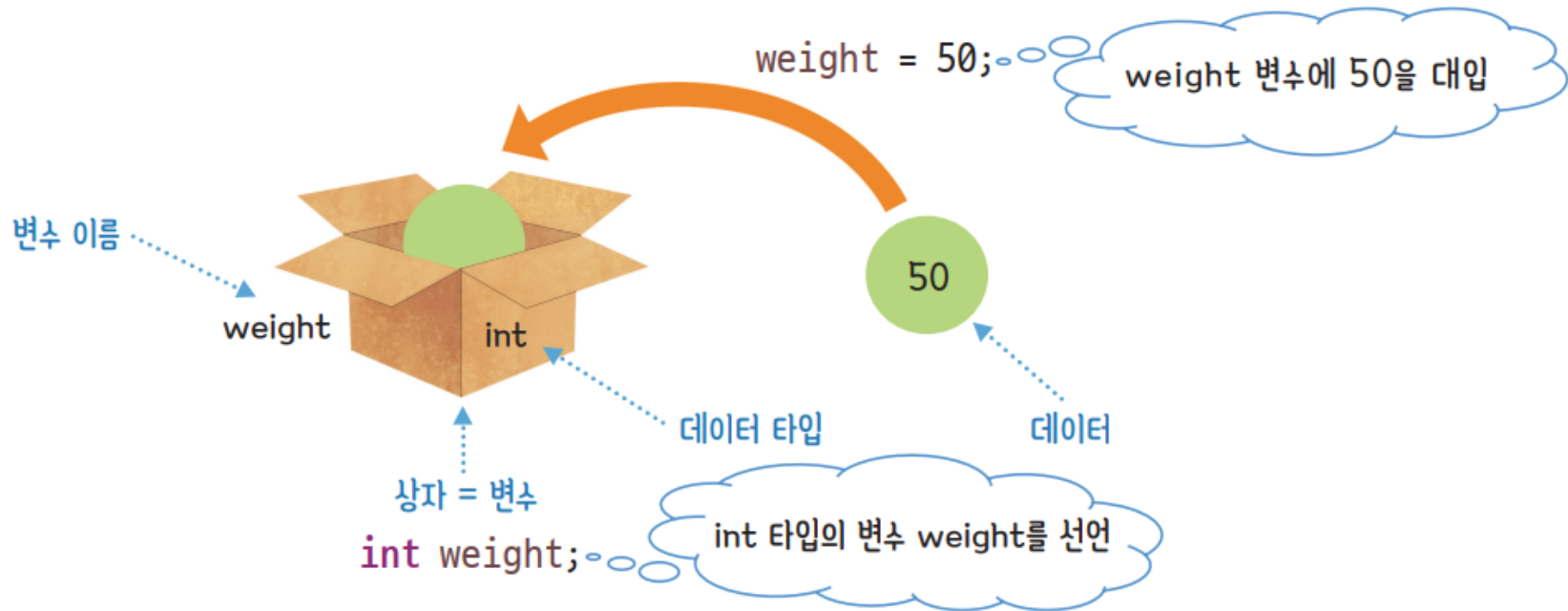
(a) 별도의 선언 및 초기화

(b) 동시에 선언 및 초기화



변수

■ 변수 사용



변수

■ var 예약어

- 자바 10부터 지원
- 초깃값을 통하여 데이터 타입 추론 가능
- 식별자로 사용 가능

```
var number = 100;    // var은 정수를 나타낼 수 있는 int 타입으로 추론한다.  
var korean = "한국"; // var은 문자열을 나타낼 수 있는 String 타입으로 추론한다.  
var oops;           // 오류
```

- 예제 : [sec03/VarDemo](#)

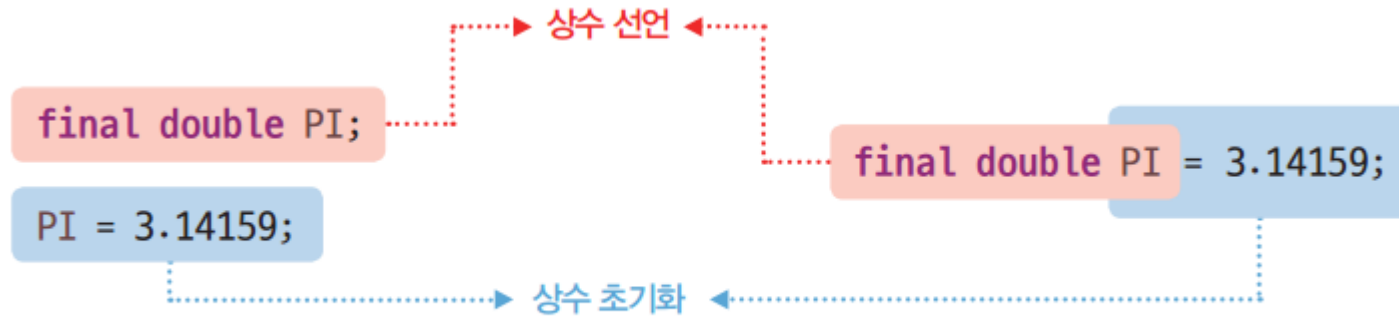
```
public class VarDemo {  
    // var a = 1;  
  
    public static void main(String[] args) {  
        int var = 1;  
        var x = 1;  
  
        // var x = 1, y = 3, z = 4;  
        // var str = null;  
        // var oops;  
        // oops = 1;  
    }  
  
    // void test(var x) { }  
}
```



변수

■ 상수

- 프로그램 실행 중 변경할 수 없는 데이터를 담는 변수
- 예를 들어 원주율 값(3.14159)이나 빛의 속도($3 \times 10^8 \text{m/s}$) 등
- 상수 이름은 변수와 구분하려고 모두 대문자로 표기
- 반드시 final 키워드로 지정



(a) 별도의 선언 및 초기화

(b) 동시에 선언 및 초기화

■ 상수와 리터럴



타입 변환

■ 자동 타입 변환

```
double d1 = 5 * 3.14; // 정수 5를 실수 5.0으로 자동 타입 변환
double d2 = 1;        // 정수 1을 실수 1.0으로 자동 타입 변환
```

■ 강제 타입 변환

```
// double의 3.14를 float로 형 변환해 f에 3.14F 저장
float f = (float)3.14;

// int의 300을 byte로 형 변환하면 데이터 손실 발생
byte b = (byte)300;

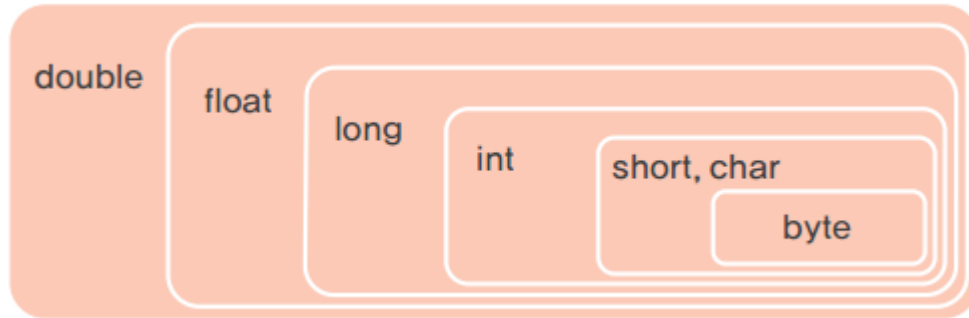
// double의 3.14를 byte로 형 변환하면 데이터가 손실되고 3만 저장
byte x = (byte)3.14;

// float의 3.14를 double로 형 변환하면 데이터 손실 없이 저장
double d = (double)3.14f;
```



타입 변환

- 연산 중 필요하면 타입 범위가 넓은 방향으로 자동 타입 변환



- 예

`i = 7 / (double) 4;`

❌

- 정수 4를 double 타입 실수 4.0으로 강제 타입 변환한다.
- 정수 7을 double 타입 실수 7.0으로 자동 타입 변환한다.
- $7.0 \div 4.0 \rightarrow 1.75$
- double 타입 1.75를 int 타입 변수 i에 저장할 수 없다.



타입 변환

■ 예제

- 코드 : [sec03/CastDemo](#)
- 실행 결과

1

1.0

1.75

byte 타입으로 변환할 수 없습니다.

```
public class CastDemo {  
    public static void main(String[] args) {  
        int i;  
        double d;  
        byte b;  
  
        i = 7 / 4;  
        System.out.println(i);  
        d = 7 / 4;  
        System.out.println(d);  
        d = 7 / (double) 4;  
        System.out.println(d);  
  
        // i = 7 / (double) 4;  
  
        i = 300;  
        if (i < Byte.MIN_VALUE || i > Byte.MAX_VALUE)  
            System.out.println("byte 타입으로 변환할 수 없습니다.");  
        else  
            b = (byte) i;  
    }  
}
```

