Revisit Java
(Variable and Inputs)

Fall, 2020

Textbook

- 명품 Java Programming (황기태, 김효수)
- https://www.booksr.co.kr/html/book/book.asp?seq=697068

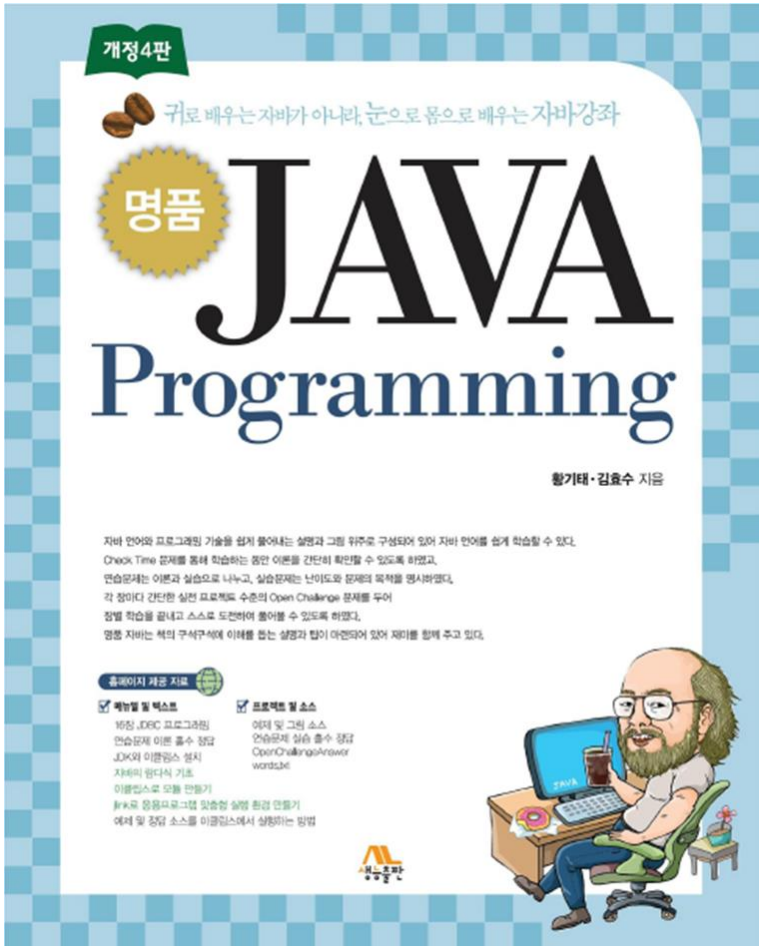


Table of Contents

- Identifier
- Keyword
- Naming Convention
- Data Type
- Literal
- Constant
- Type Coercion
- Standard Input (Scanner)

Identifier

- Identifier
  - Given names for classes, variables, constants, methods
- Naming rules for identifiers
  - No allowed
    - Special characters such as ‘@’, ‘#’, ‘!’ except ‘_’, ‘$’
    - Tab, White Space
    - Pre-defined keywords (e.g., class)
    - Word starts with number (e.g., 1st)
  - Allowed
    - Unicode characters including all the alphabetical characters, and even Korean
    - No length limitation
  - Case sensitive
    - Test is different from test

Identifier: example

- Examples

```
int     name;
char    student_ID;                    // '_' 사용 가능
void    $func() { }                    // '$' 사용 가능
class   Monster3 { }                   // 숫자 사용 가능
int     whatsyournamemynameiskitae;    // 길이 제한 없음
int     barChart; int barchart;        // 대소문자 구분. barChart와 barchart는 다름
int     가격;                          // 한글 이름 사용 가능
```

- Counter Examples

```
int     3Chapter;                      // 식별자의 첫문자로 숫자 사용 불가
class   if { }                         // 자바의 예약어 if 사용 불가
char    false;                         // false 사용 불가
void    null() { }                     // null 사용 불가
class   %calc { }                      // '%'는 특수문자
```

Pre-defined Keywords

abstract	continue	for	new	switch
assert	default	if	package	synchronized
boolean	do	goto	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Naming Convention

- Readability
  - Get to know the purpose: sum instead of s
  - Full name instead of Abbreviation: AutoVendingMachine instead of AVM
- Hungarian naming convention
  - Naming for Class
    - Starts with Capital character
    - Capital character at each word
  - Naming for Variable and Method
    - Start with Non-capital character
    - Capital character at each word
  - Naming for constants
    - Capitalize each character

```
public class HelloWorld { }
class AutoVendingMachine { }
```

```
int myAge;
boolean isSingle;
public int getAge() {}
```

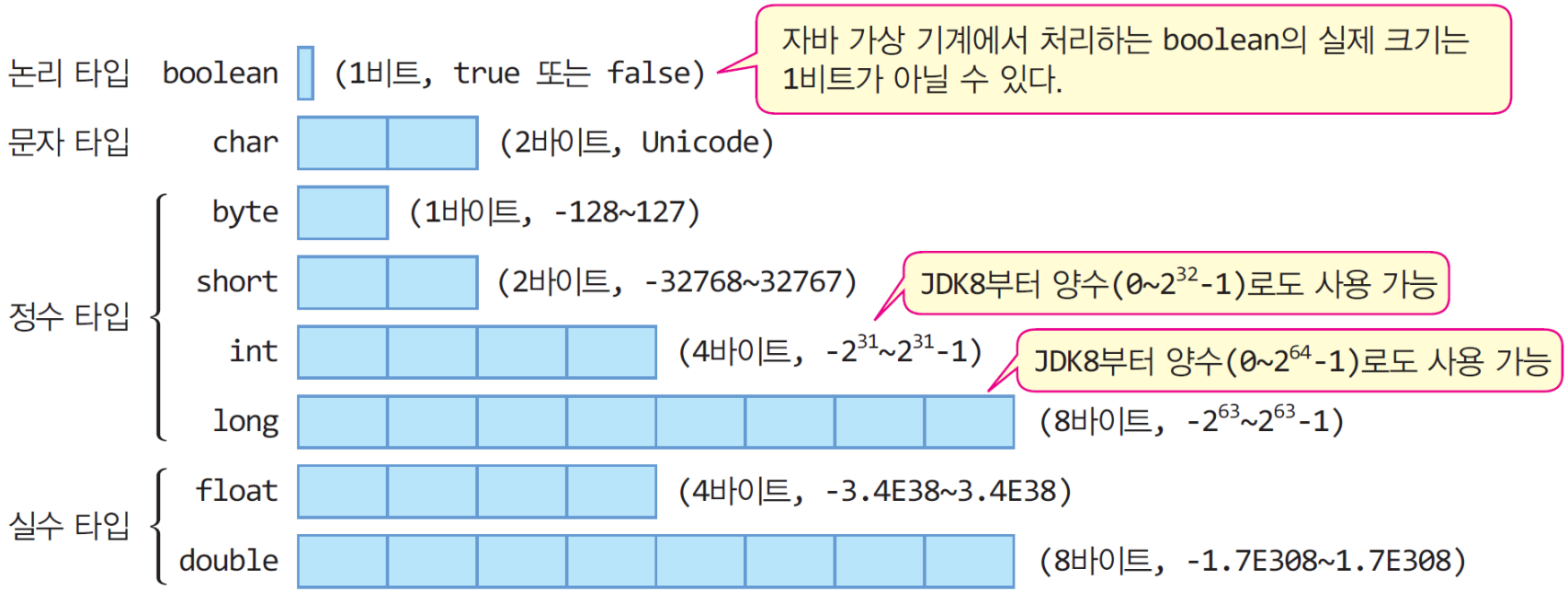
```
final static double PI = 3.141592;
```

Data Type

- Primitive Type (8 units)
  - boolean
  - char
  - byte
  - short
  - int
  - long
  - float
  - double
- Reference Type
  - Reference for Array
  - Reference for Class
  - Reference for Interface

Data Type: Primitive Type

- Feature
  - Size for each type is fixed regardless of the type of CPU and OS



Data Type: String (Non-primitive type)

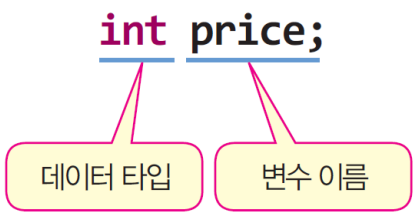
- Reference type (Non-primitive type)
- Much more convenient than an array of characters
  - char[] hello = {‘H’,’e’,’l’,’l’,’o’};

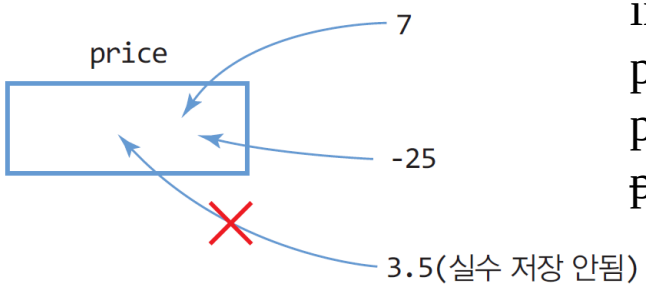
H	e	l	l	o

    - What if adding “World” to hello
      - Declare another array with size 11, and make the word
    - How to read the array
      - Iteration
  - String hello = “Hello”;
    - Method overriding for +
      - hello = hello + “ World”;

Variable

- Variable
  - A memory location to store temporary value
  - Modifiable in run-time
- How to declare?
  - type \s variable identifier





int price;
price = 7;
price = price – 32;
~~price = 3.5;~~ // compile error

Variable Declaration

- Example

```
int radius;
char c1, c2, c3; // 3 개의 변수를 한 번에 선언한다.
```

What is this called?

```
int radius = 10;
char c1 = 'a', c2 = 'b', c3 = 'c';
double weight = 75.56;
```

```
radius = 10 * 5;
c1 = 'r';
weight = weight + 5.0;
```

Literal

- Self-expressive specific values in your program
  - Example
    - Integer value: 5
    - Real value: 3.5
    - Character: ‘c’
    - Logical value: true
    - String: “hello”

String helloWorld = “Hello World!”;

Data Type | Identifier | Literal

Literal: Integer

- Integer Literal
  - Decimal number
    - int n = 15;
  - Octal Number
    - int m = 015; // start with 0
  - Hexadecimal
    - int k = 0x15; // start with 0x
  - Binary number
    - int b = 0b0101; // start with 0b
- Long Literal
  - Can express larger number than Integer
    - long g = 24L; // end with L or l

Literal: Real value

- Floating number or Exponent
  - Example
    - 12. 12.0 .1234 0.1234 1234E-4

- Floating Number
  - float or double
  - Example
    - double d = 0.1234;
    - double e = 1234E-4; // $1234 * 10^{-4}$
    - float f = 0.1234f; // End with f or F (more formal)
    - Double w = .1234D; // End with d or D (more formal)

Literal: Character

- Embrace with ‘’
  - Example
    - ‘w’, ‘A’, ‘가’, ‘*’, '3', '글', \u0041
      - \u다음에 4자리 16진수(2바이트의 유니코드)
      - \u0041 -> 문자 'A'의 유니코드(0041)
      - \uae00 -> 한글문자 '글'의 유니코드(ae00)

- Special Characters
  - Start with back slash (\)

종류	의미	종류	의미
'\b'	백스페이스(backspace)	'\r'	캐리지 리턴(carriage return)
'\t'	탭(tab)	'\"'	이중 인용부호(double quote)
'\n'	라인피드(line feed)	'\''	단일 인용부호(single quote)
'\f'	폼피드(form feed)	'\\'	백슬래시(backslash)

Literal: Logical value

- true, false
  - Assigned as a value of boolean type variable
  - Utilized in conditional statements

```
boolean a = true;
boolean b = 10 > 0; // 10>0가 참이므로 b 값은 true
boolean c = 1; // 타입 불일치 오류. C/C++와 달리 자바에서 1,0을 참,
거짓으로 사용 불가

while(true) { // 무한 루프. while(1)로 사용하면 안 됨
...
}
```

Literal: Miscellaneous

- null literal
  - means ‘empty’
  - Used for reference types

- String Literal
  - Embrace with “”
    - e.g., "Good", "Morning", "자바", "3.19", "26", "a"

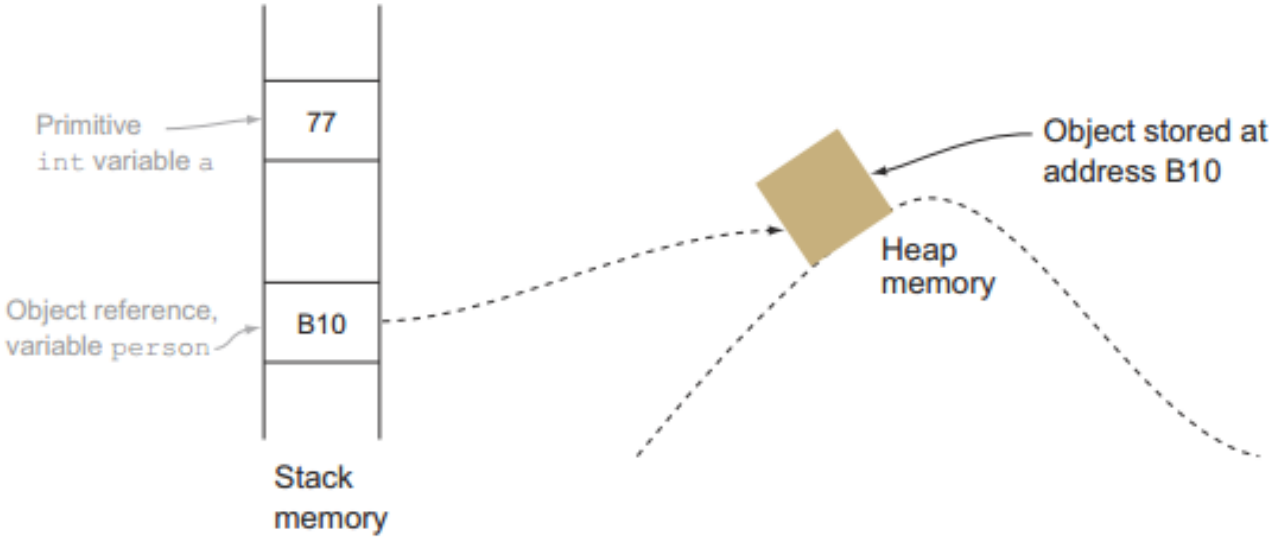
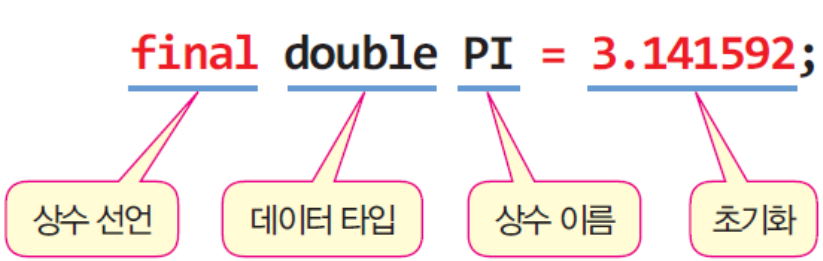


Figure 2.13 Primitive variables store the actual values, whereas object reference variables store the addresses of the objects they refer to.

Constant Value

- Constant Value
  - By using final keyword
  - Initialize its value at declaration
  - Unmodifiable



Practice 1

- Practice declaration of primitive types

```java
public class P1PrimitiveTypes {

        public static void main(String[] args) {

                // Java provides 8 primitive types
                // TYPE IDENTIFIER then Assignment value
                // boolean is yes or no
                boolean isMale = true;
                // char is one character
                char bloodType = 'A';
                // byte is one byte integer
                byte semester = 3;
                // short is two byte integer

                // int is four byte integer

                // long is 8 byte integer

                // float is 4 byte real number

                // double is 8 byte real number

                System.out.println("Are you male? " + isMale);
                System.out.println("Your blood type? " + bloodType);
                System.out.println("How much semesters have you done?" + (semester - 1));
                // Practice others
        }
}
```

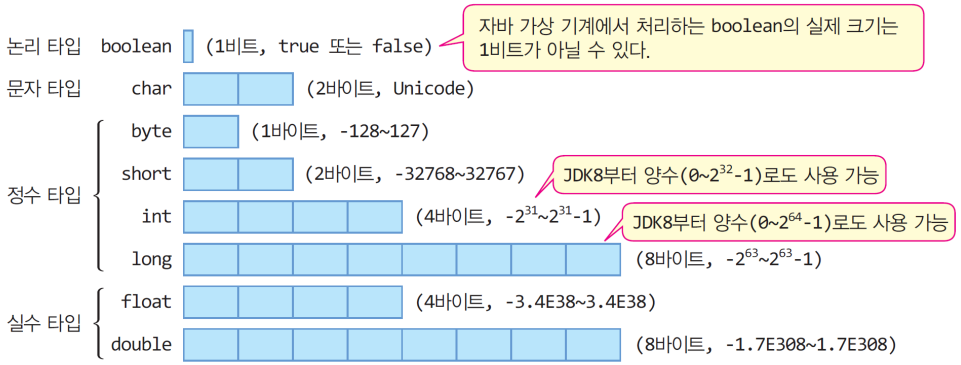
Practice 2

- Practice declaration of String type (Reference Type)

```
3   public class P2StringType {
4
5           public static void main(String[] args) {
6
7                   // String is not a primitive type
8                   // Very convenient compared to Char array in C++
9                   String name = "YOUR NAME";
10
11                  // void java.io.PrintStream.println(String x)
12                  // println methods print out String variable
13                  // Other types would be 'converted' into String
14                  System.out.println(name);
15
16          }
17  }
```

Type Coercion: Automatic

- Type Coercion
  - The conversion from a value of one literal to another value of another literal
    - Example
      - long longValue = 3; // 3 is integer
      - float floatValue = 3; // 3 is integer
  - Automatic type conversion
    - Conducted by compiler
    - From short size value to long size one



    - Literal types in an expression are mismatched

```
double d = 3.14 * 10;      // 실수 연산을 하기 위해 10이 10.0으로 자동 변환
                           // 다른 피연산자 3.14가 실수이기 때문
```

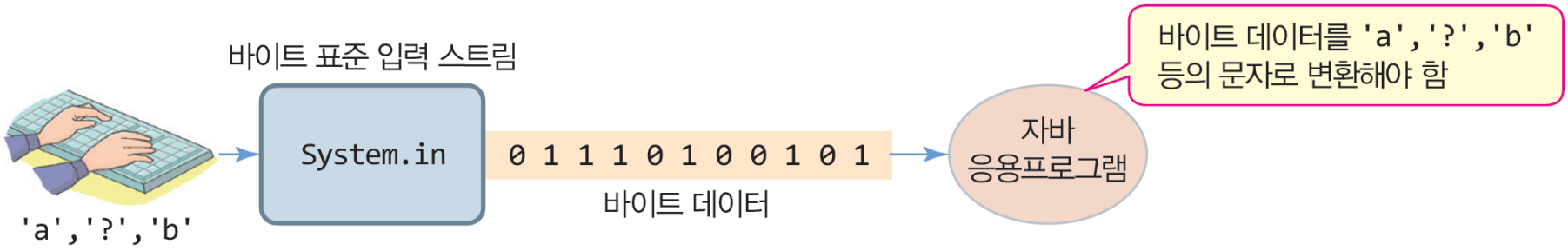
Practice 3

- Practice Type Coercion

```java
public class P3TypeCasting {

        public static void main(String[] args) {

                // Automatic Type Casting
                // Think about a water basket

                // byte: 1 byte integer
                // int: 4 byte integer

                // 1 byte = 8 bits -> 2^8 -> -128 ~ 127 right?
                byte x = 127;
                // You will see the error
                // byte y = 128;
                System.out.println(x);

                // byte to int is available
                int y = x;
                System.out.println(y);

                // Manual Type Casting
                int a = 127;
                // a reversed way is not possible by compiler
                // byte b = a;
                // A programmer should guarantee there is no problem
                byte b = (byte) a;
                System.out.println(b);
                // For this situation
                int c = 123123123;
                byte d = (byte) c;
                System.out.println(d);
        }
}
```

Standard Input (Scanner)

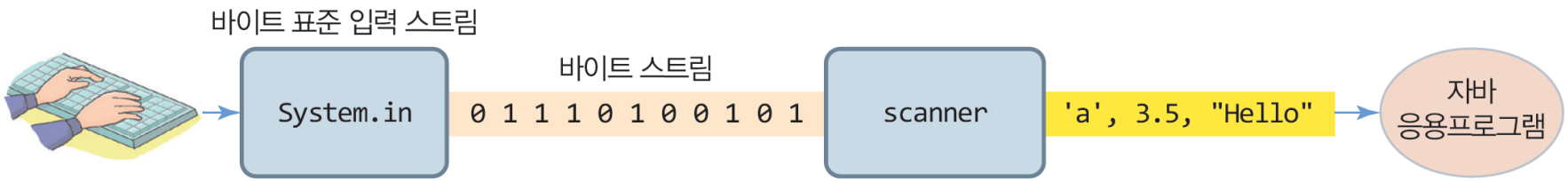
- System.in
  - Standard Input from Keyboard
  - returns key inputs as bytes



Standard Input (Scanner)

- Scanner Class
  - Let System.in read keyboard inputs
  - Return either character, integer, real number, boolean, or String from the inputs
- Instantiate the class
  - java.util.Scanner; will be used (Verbose, you can shorten it via import)

```
import java.util.Scanner; // import for concise code
...
Scanner a = new Scanner(System.in); // instantiate Scanner class
```

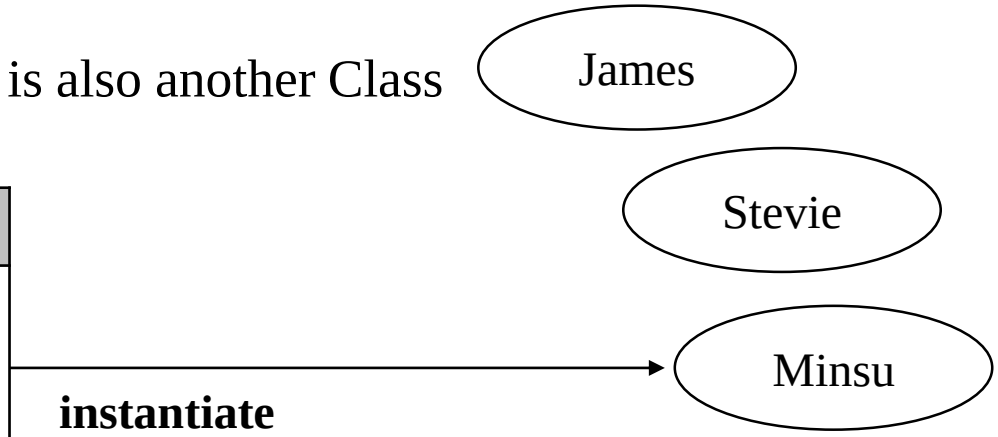


Standard Input (Scanner)

- Basic introduction to Class
  - java.util.Scanner is a Class
  - your kr.ac.sejong.icse.HellWorld is also another Class

Human Class	
Local Variables	float height; float weight; boolean gender; …

Methods	void setWeight(float weight);
	float getWeight();
	void shout();



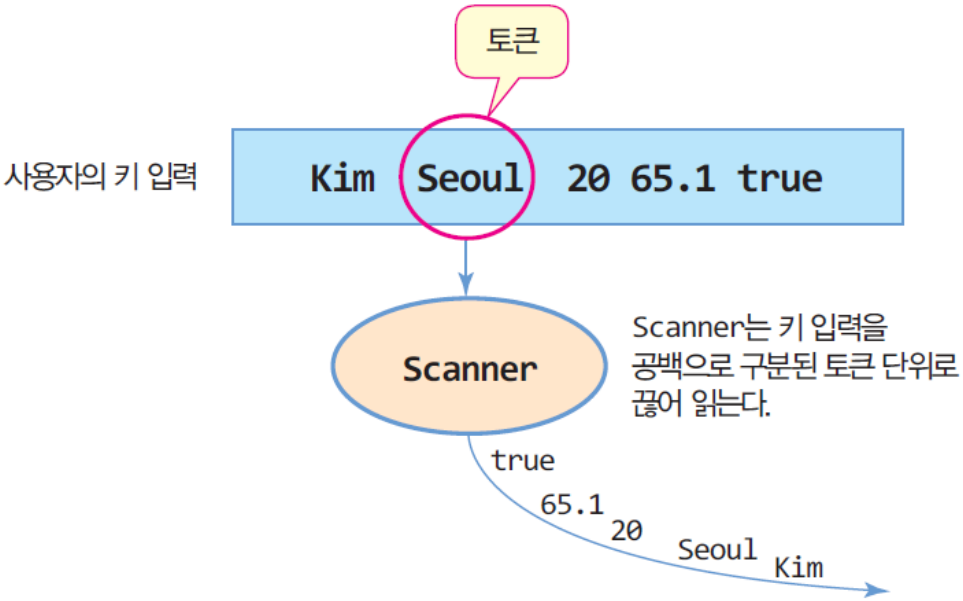
instantiate

```
Human minsu = new Human();
minsu.setWeight(70.5);
int minsuWeight = minsu.getWeight();
```

- You can use Class directly via static variables and static methods
  - Learn it later

Standard Input (Scanner)

- Keyboard Inputs
  - Scanner delimits inputs with
    - Tab: ‘\t’
    - New line: ‘\n’
    - White Space: ‘ ‘
    - etc.




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

String name = scanner.next();                  // "Kim"
String city = scanner.next();                  // "Seoul"
int age = scanner.nextInt();                   // 20
double weight = scanner.nextDouble();          // 65.1
boolean single = scanner.nextBoolean();        // true
```

Standard Input (Scanner)

메소드	설명
String next()	다음 토큰을 문자열로 리턴
byte nextByte()	다음 토큰을 byte 타입으로 리턴
short nextShort()	다음 토큰을 short 타입으로 리턴
int nextInt()	다음 토큰을 int 타입으로 리턴
long nextLong()	다음 토큰을 long 타입으로 리턴
float nextFloat()	다음 토큰을 float 타입으로 리턴
double nextDouble()	다음 토큰을 double 타입으로 리턴
boolean nextBoolean()	다음 토큰을 boolean 타입으로 리턴
String nextLine()	'\n'을 포함하는 한 라인을 읽고 '\n'을 버린 나머지 문자열 리턴
void close()	Scanner의 사용 종료
boolean hasNext()	현재 입력된 토큰이 있으면 true, 아니면 입력 때까지 무한정 대기, 새로운 입력이 들어올 때 true 리턴. crtl-z 키가 입력되면 입력 끝이므로 false 리턴

Practice 4

- Practice Scanner

```java
import java.util.Scanner;

public class P4Scanner {

        public static void main(String[] args) {
                // Scanner class enables you to get standard input ( your keyboard )
                Scanner scanner = new Scanner(System.in);
                // Scanner.next() method waits your input with the delimeter ' '
                System.out.print("What is your name? ");
                String name = scanner.next();
                System.out.println(name);
                scanner.close();
        }
}
```

Practice 5

- Practice Scanner

```java
import java.util.Scanner;

// Textbook
public class P5WrapUp {

        public static void main(String[] args) {
                // Write a program
                // Ask name, city, age, weight, isSingle and print out
                System.out.println("이름, 도시, 나이, 체중, 독신 여부를 빈칸으로 분리하여 입력하세요");
                Scanner scanner = new Scanner(System.in);
                String name = scanner.next(); // 문자열 읽기
                System.out.print("이름은 " + name + ", ");
                scanner.close();
        }
}
```

Summary

- Programming Basic
  - Identifier
  - Keyword
  - Naming Convention
  - Data Type
  - Literal
  - Constant
  - Type Conversion
  - Standard Input (Scanner)