

자바 기본 프로그래밍

컴퓨터공학전공
박요한

■ 수업 내용

- 자바 프로그램의 구조
- 식별자
- 자바의 데이터 타입
- 자바에서의 키 입력
- 연산
- 조건문

예제 2-1 자바프로그램의 기본 구조

```
/*  
* 소스 파일 : Hello.java  
*/
```

```
public class Hello {
```

```
    public static int sum(int n, int m) {  
        return n + m;  
    }
```

메소드

```
// main() 메소드에서 실행 시작
```

```
public static void main(String[] args) {  
    int i = 20;  
    int s;  
    char a;
```

```
    s = sum(i, 10); // sum() 메소드 호출
```

```
    a = '?';
```

```
    System.out.println(a); // 문자 '?' 화면 출력
```

```
    System.out.println("Hello"); // "Hello" 문자열 화면 출력
```

```
    System.out.println(s); // 정수 s 값 화면 출력
```

```
}
```

```
}
```

클래스

메소드

```
?  
Hello  
30
```

예제 2-1 코드 설명

■ 클래스 만들기

- ✓ Hello 이름의 클래스 선언

```
public class Hello {  
}
```

- ✓ class 키워드로 클래스 선언
- ✓ public 선언하면 다른 클래스에서 접근 가능
- ✓ 클래스 코드는 {} 내에 모두 작성

■ 주석문

- ✓ // 한 라인 주석
- ✓ /* 여러 행 주석 */

■ main() 메소드

- ✓ 자바 프로그램은 main()에서 실행 시작

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

- ✓ public static void으로 선언
- ✓ String[] args로 실행 인자를 전달 받음

■ 메소드

- ✓ C/C++에서의 함수를 메소드로 지칭

```
public static int sum(int n, int m) {  
    ...  
}
```

- ✓ 클래스 바깥에 작성할 수 없음

■ 메소드 호출

- ✓ sum() 메소드 호출

```
int i = 20;  
s = sum(i, 10);
```

- ✓ sum() 호출 시 변수 i의 값과 정수 10을 전달
- ✓ sum()의 n, m에 각각 20, 10 값 전달
- ✓ sum()은 n과 m 값을 더한 30 리턴
- ✓ 변수 s는 정수 30을 전달받음

예제 2-1 설명 (계속)

■ 변수 선언

- ✓ 변수 타입과 변수 이름 선언

```
int i=20;  
char a;
```

- ✓ 메소드 내에서 선언된 변수는 지역 변수
 - Ⓢ 지역 변수는 메소드 실행이 끝나면 자동 소멸

■ 문장

- ✓ ;로 한 문장의 끝을 인식

```
int i=20;  
s = sum(i, 20);
```

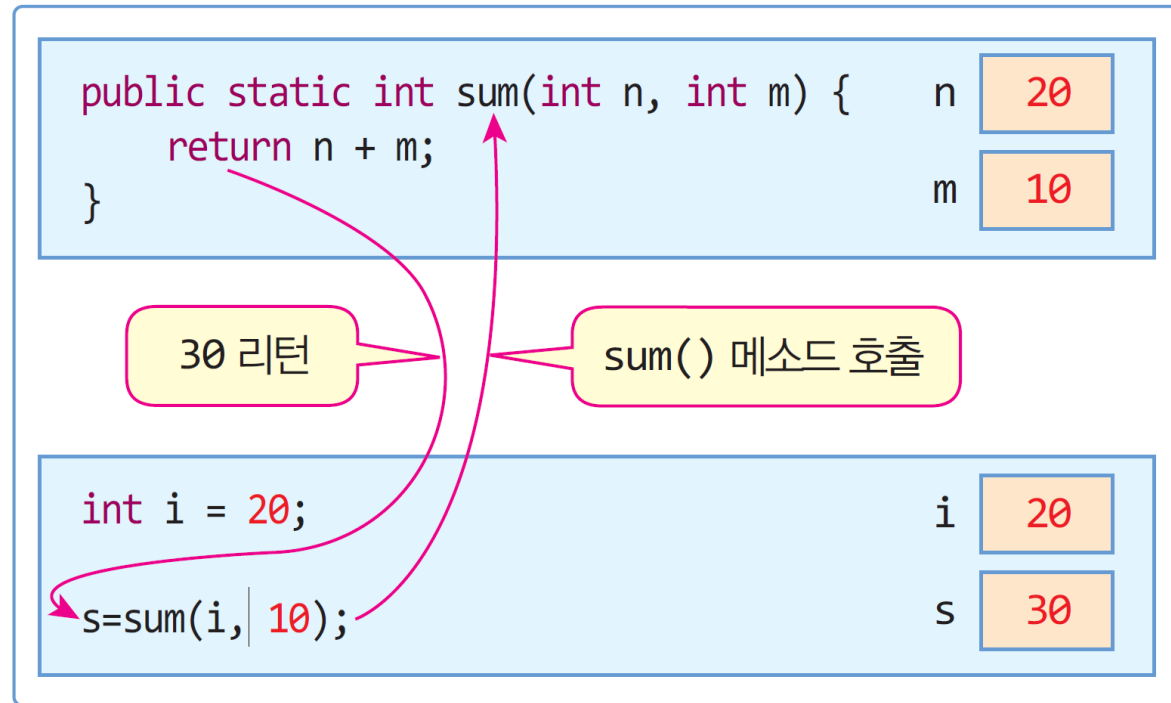
■ 화면 출력

- ✓ 표준 출력 스트림에 메시지 출력

```
System.out.println("Hello"); // "Hello" 화면 출력
```

- ✓ 표준 출력 스트림 System.out의 println() 메소드 호출
- ✓ println()은 여러 타입의 데이터 출력 가능
- ✓ println()은 출력 후 다음 행으로 커서 이동

sum() 메소드 호출과 리턴



클래스 Hello

■ 식별자 (identifier)

- 식별자란?
 - ✓ 클래스, 변수, 상수, 메소드 등에 붙이는 이름
- 식별자의 원칙
 - ✓ '@', '#', '!'와 같은 특수 문자, 공백 또는 탭은 식별자로 사용할 수 없으나 '-', '\$'는 사용 가능
 - ✓ 유니코드 문자 사용 가능. 한글 사용 가능
 - ✓ 자바 언어의 키워드는 식별자로 사용불가
 - ✓ 식별자의 첫 번째 문자로 숫자는 사용불가
 - ✓ '-' 또는 '\$'를 식별자 첫 번째 문자로 사용할 수 있으나 일반적으로 잘 사용하지 않는다.
 - ✓ 불린 리터럴 (true, false)과 널 리터럴(null)은 식별자로 사용불가
 - ✓ 길이 제한 없음
- 대소문자 구별
 - ✓ Test와 test는 별개의 식별자

식별자 이름 사례

■ 사용 가능한 예

```
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
```

■ 잘못된 예

```
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
```


자바 키워드

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

좋은 이름 붙이는 언어 관습

- 기본 : 가독성 높은 이름
 - ✓ 목적을 나타내는 이름 붙이기 : s 보다 sum
 - ✓ 충분히 긴 이름으로 붙이기 : AVM보다 AutoVendingMachine
- 자바 언어의 이름 붙이는 관습 : 헝가리언 이름 붙이기

- ✓ 클래스 이름

- ⌚ 첫 번째 문자는 대문자로 시작
- ⌚ 각 단어의 첫 번째 문자만 대문자

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

- 변수, 메소드 이름

- ⌚ 첫 단어 이후 각 단어의 첫 번째 문자는 대문자로 시작

- 상수 이름

- ⌚ 모든 문자를 대문자로 표시

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

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

자바의 데이터 타입

■ 자바의 데이터 타입

✓ 기본 타입 : 8 개

- ② boolean
- ② char
- ② byte
- ② short
- ② int
- ② long
- ② float
- ② double

✓ 레퍼런스 타입 : 1 개이며 용도는 다음 3 가지

- ② 배열(array)에 대한 레퍼런스
- ② 클래스(class)에 대한 레퍼런스
- ② 인터페이스(interface)에 대한 레퍼런스

자바의 기본 타입

■ 특징

✓ 기본 타입의 크기가 정해져 있음

Ⓢ CPU나 운영체제에 따라 변하지 않음

논리 타입 boolean (1비트, true 또는 false)

자바 가상 기계에서 처리하는 boolean의 실제 크기는 1비트가 아닐 수 있다.

문자 타입 char (2바이트, Unicode)

정수 타입 { byte (1바이트, -128~127)

short (2바이트, -32768~32767)

JDK8부터 양수($0 \sim 2^{32}-1$)로도 사용 가능

int (4바이트, $-2^{31} \sim 2^{31}-1$)

JDK8부터 양수($0 \sim 2^{64}-1$)로도 사용 가능

long (8바이트, $-2^{63} \sim 2^{63}-1$)

실수 타입 { float (4바이트, $-3.4\text{E}38 \sim 3.4\text{E}38$)

double (8바이트, $-1.7\text{E}308 \sim 1.7\text{E}308$)

문자열

- 문자열은 기본 타입이 아님
- String 클래스로 문자열 표현
 - ✓ 문자열 리터럴 - “JDK”, “한글”, “계속하세요”

```
String toolName="JDK";
```

- ✓ 문자열이 섞인 + 연산 -> 문자열 연결

```
toolName + 1.8          -> "JDK1.8"  
"(" + 3 + "," + 5 + ")" -> "(3,5)"  
System.out.println(toolName + "이 출시됨"); // "JDK1.8이 출시됨" 출력
```

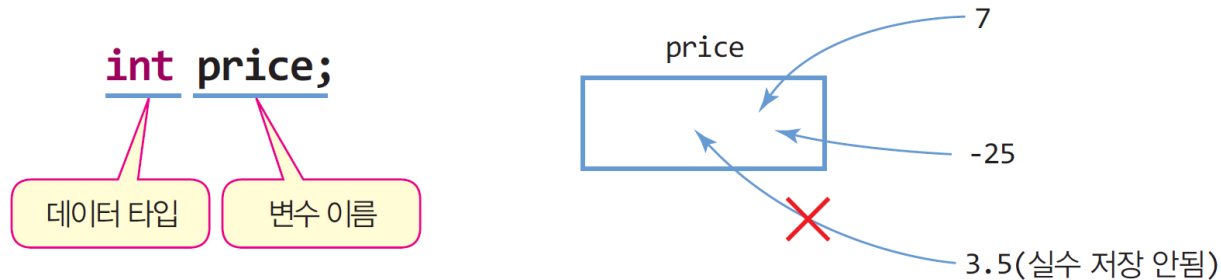
■ 변수와 선언

■ 변수

- ✓ 프로그램 실행 중에 값을 임시 저장하기 위한 공간
 - Ⓢ 변수 값은 프로그램 수행 중 변경될 수 있음
- ✓ 데이터 타입에서 정한 크기의 메모리 할당

■ 변수 선언

- ✓ 변수의 타입 다음에 변수 이름을 적어 변수를 선언



■ 변수 선언 사례

■ 변수 선언 사례

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

■ 변수 선언과 초기화

✓ 선언과 동시에 초기값 지정

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

■ 변수 읽기와 저장

✓ 대입 연산자인 = 다음에 식(expression)

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

리터럴과 정수 리터럴

■ 리터럴(literal)

- ✓ 프로그램에서 직접 표현한 값
- ✓ 정수, 실수, 문자, 논리, 문자열 리터럴 있음
 - ☞ 사례) 34, 42.195, ' % ', true, " hello "

■ 정수 리터럴

- ✓ 10진수, 8진수, 16진수, 2진수 리터럴

15	-> 10진수 리터럴 15
0 15	-> 0으로 시작하면 8진수. 십진수로 13
0x 15	-> 0x로 시작하면 16진수. 십진수로 21
0b 0101	-> 0b로 시작하면 2진수. 십진수로 5



```
int n = 15;  
int m = 015;  
int k = 0x15;  
int b = 0b0101;
```

- ✓ 정수 리터럴은 int 형으로 컴파일
- ✓ long 타입 리터럴은 숫자 뒤에 L 또는 l을 붙여 표시
 - ☞ ex) long g = 24L;

실수 리터럴

- ✓ 소수점 형태나 지수 형태로 표현한 실수

Ⓢ 12. 12.0 .1234 0.1234 1234E-4

- ✓ 실수 타입 리터럴은 double 타입으로 컴파일

```
double d = 0.1234;  
double e = 1234E-4; // 1234E-4 = 1234x10-4이므로 0.1234와 동일
```

- ✓ 숫자 뒤에 f(float)나 d(double)을 명시적으로 붙이기도 함

```
float f = 0.1234f;  
double w = .1234D; // .1234D와 .1234는 동일
```

문자 리터럴

✓ 단일 인용부호(' ')로 문자 표현

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


```
char a = 'A';  
char b = '글';  
char c = \u0041; // 문자 'A'의 유니코드 값(0041) 사용  
char d = \uae00; // 문자 '글'의 유니코드 값(ae00) 사용
```

✓ 특수문자 리터럴은 백슬래시(\)로 시작

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)

논리 리터럴과 boolean 타입

- 논리 리터럴은 2개뿐
 - ✓ true, false
 - ✓ boolean 타입 변수에 치환하거나 조건문에 이용

 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
...
}
```

Tip: 기본 타입 이외의 리터럴

■ null 리터럴

- ✓ 레퍼런스에 대입 사용



```
int n = null; // 기본 타입에 사용 불가  
String str = null;
```

■ 문자열 리터럴(스트링 리터럴)

- ✓ 이중 인용부호로 묶어 표현
 - ☞ 사례) "Good", "Morning", "자바", "3.19", "26", "a"
- ✓ 문자열 리터럴은 String 객체로 자동 처리

```
String str = "Good";
```

Tip. var 키워드를 사용하여 변수 타입 생략

■ var 키워드

- ✓ Java 10부터 도입된 키워드

 - Ⓢ var와 동일한 기능으로 C++(2011년 표준부터)의 auto 키워드

- ✓ 지역 변수의 선언에만 사용

- ✓ 변수 타입 선언 생략 : 컴파일러가 변수 타입 추론

■ 사용 예

```
var price = 200;           // price는 int 타입으로 결정
var name = "kitae";        // name은 String 타입으로 결정
var pi = 3.14;             // pi는 double 타입으로 결정
var point = new Point();   // point는 Point 타입으로 결정(4장 참조)
var v = new Vector<Integer>(); // v는 Vector<integer> 타입으로 결정(7장 참조)
```

■ 변수 선언문에 반드시 초깃값 지정

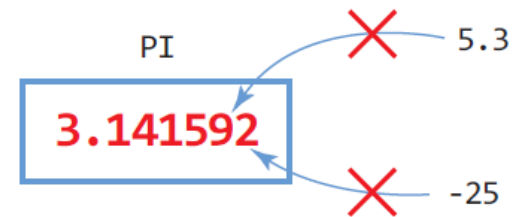
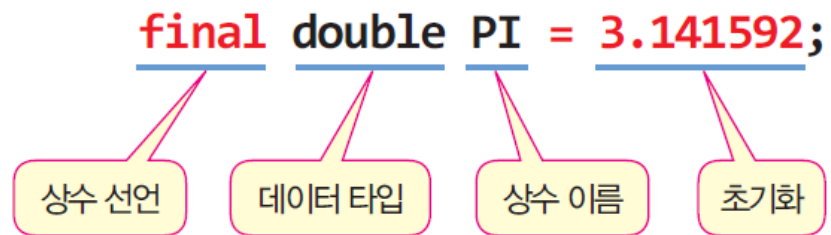


```
var name;           // 컴파일 오류. 변수 name의 타입을 추론할 수 없음
```

상수

■ 상수 선언

- ✓ final 키워드 사용
- ✓ 선언 시 초기값 지정
- ✓ 실행 중 값 변경 불가



예제 2-2 : 변수, 리터럴, 상수 활용

원의 면적을 구하는 프로그램을 작성해보자.

```
public class CircleArea {  
  
    public static void main(String[] args) {  
        final double PI = 3.14; // 원주율을 상수로 선언  
  
        double radius = 10.0; // 원의 반지름  
        double circleArea = radius*radius*PI; // 원의 면적 계산  
  
        // 원의 면적을 화면에 출력한다.  
        System.out.println("원의 면적 = " + circleArea);  
    }  
}
```

원의 면적 = 314.0