

# COMP217

## Java Programming

### Summer 2018

*Day 1*

*Introduction to computers, Java, and variables*

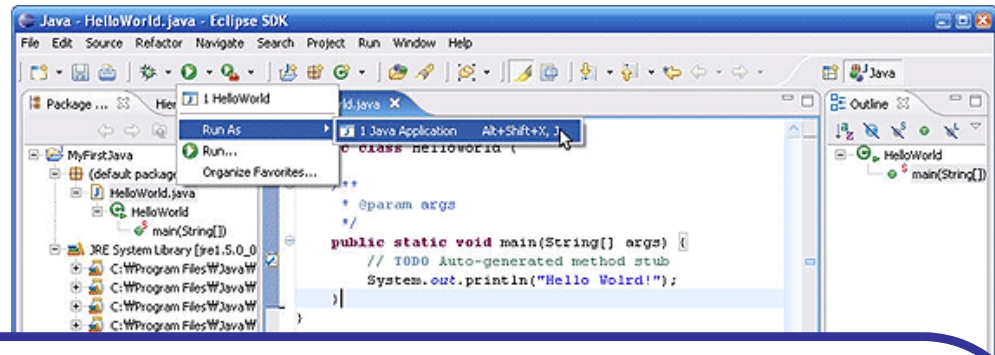
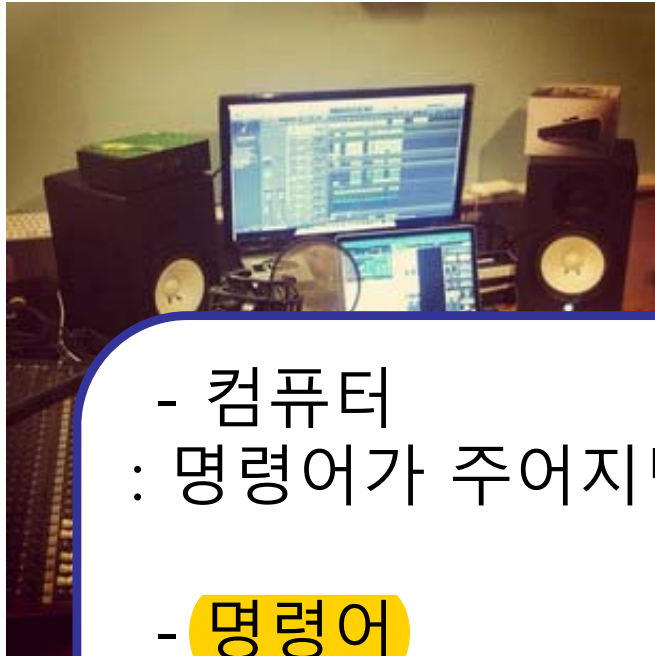
# Goals

- 이번 주에 배우게 될 내용 :
  - 프로그래밍이란?
  - 자바 프로그램 환경 구축
  - 간단한 자바 프로그램 작성
  - 변수
  - 기초 자료형
  - Scanner 클래스

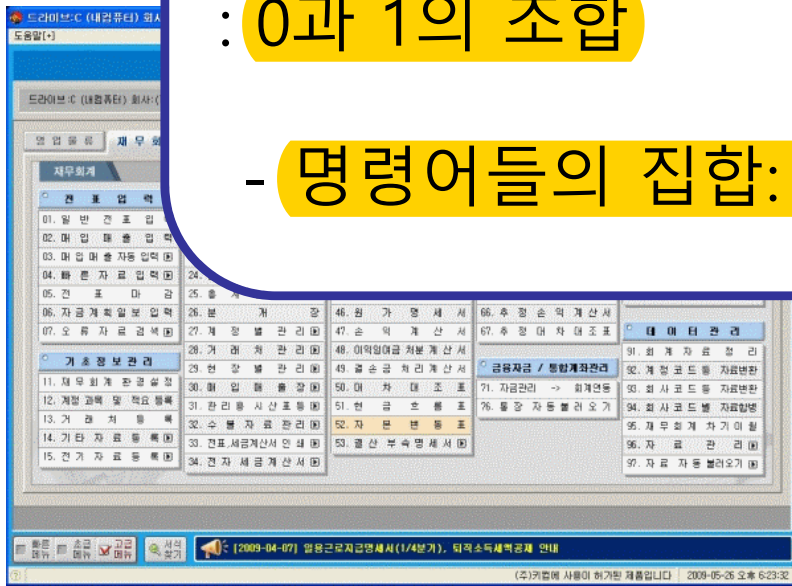
# 여러 가지 작업 도구들

작업도구	직업
피아노, 바이올린, 첼로...	

# 컴퓨터의 여러 활용들 Why?



- 컴퓨터  
: 명령어가 주어지면 그것을 처리하도록 만들어진 기계
- 명령어  
: 0과 1의 조합
- 명령어들의 집합: 프로그램

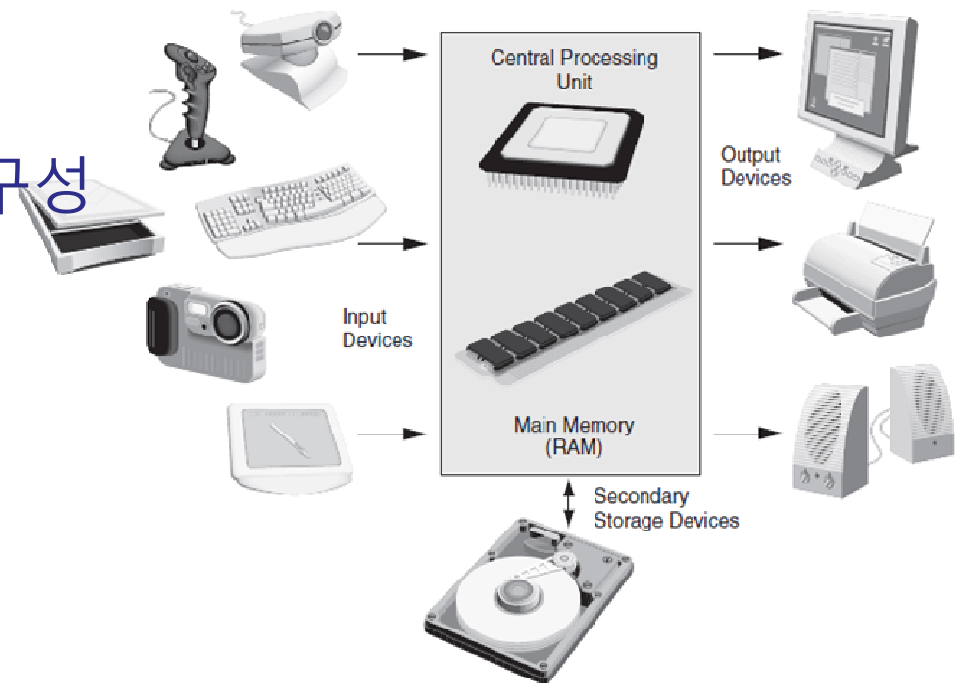


# Computer

- 하드웨어(Hardware)와 소프트웨어(Software)로 구성

- 하드웨어

- 전자회로와 물리적인 장치로 구성
- Central Processing Unit (CPU)
- Main memory (RAM)
- Secondary storage devices
- Input and Output Devices

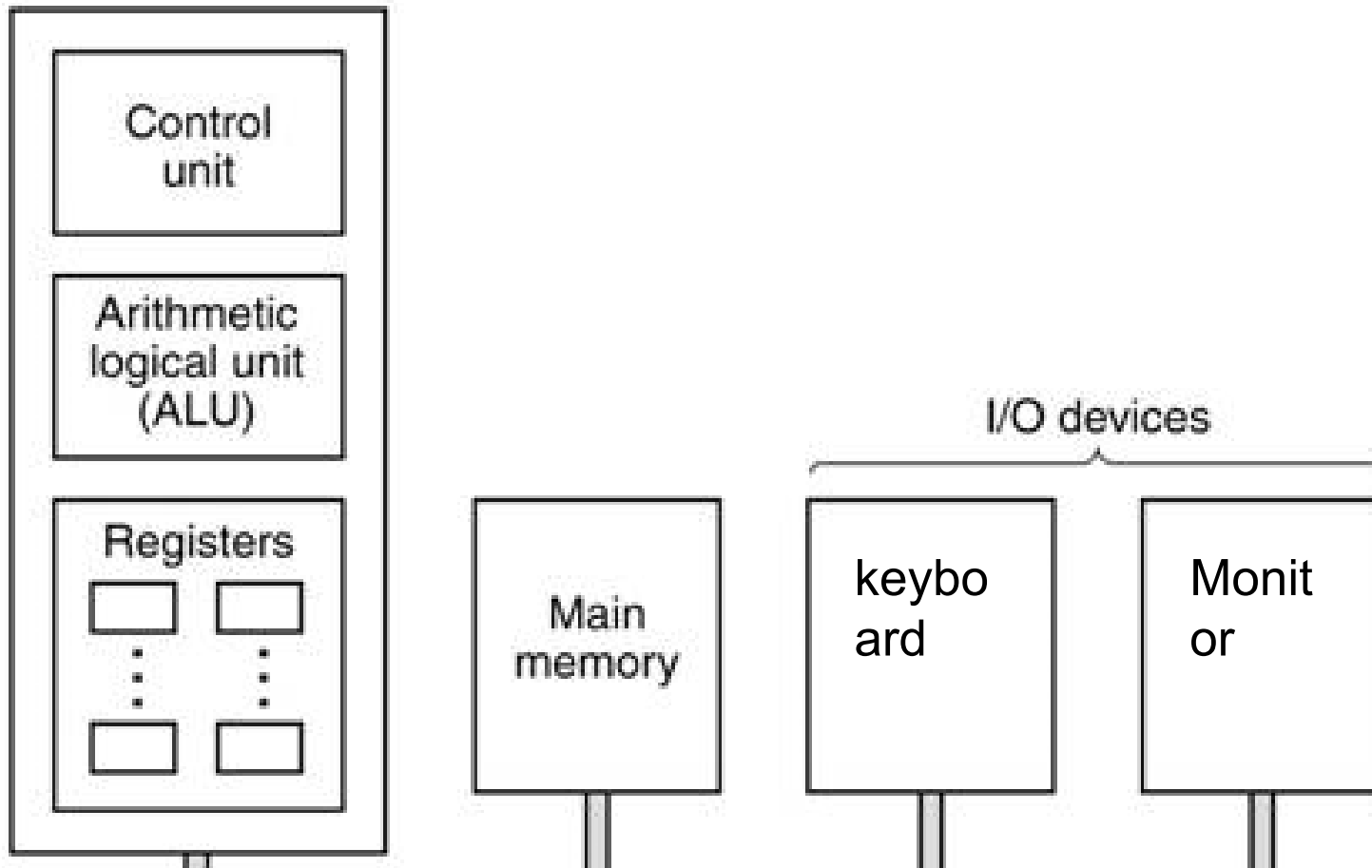


- 소프트웨어

- 하드웨어를 관리하거나, 컴퓨터에서 실행되는 프로그램
- Operating Systems
- Application Software

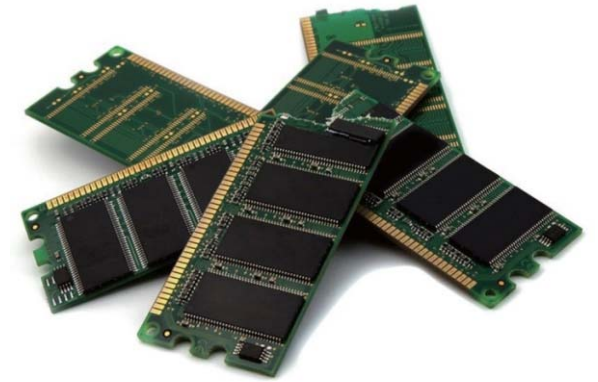
# 하드웨어

Central processing unit (CPU)

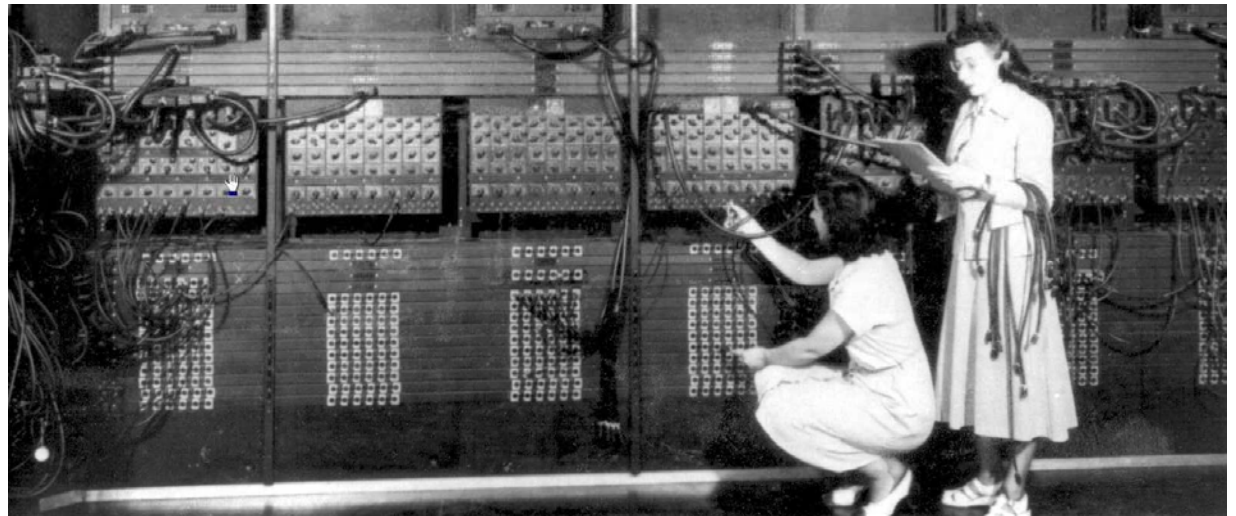


# Hardware - main memory

- RAM(random-access memory)으로 불림
- RAM에 저장되는 것은 :

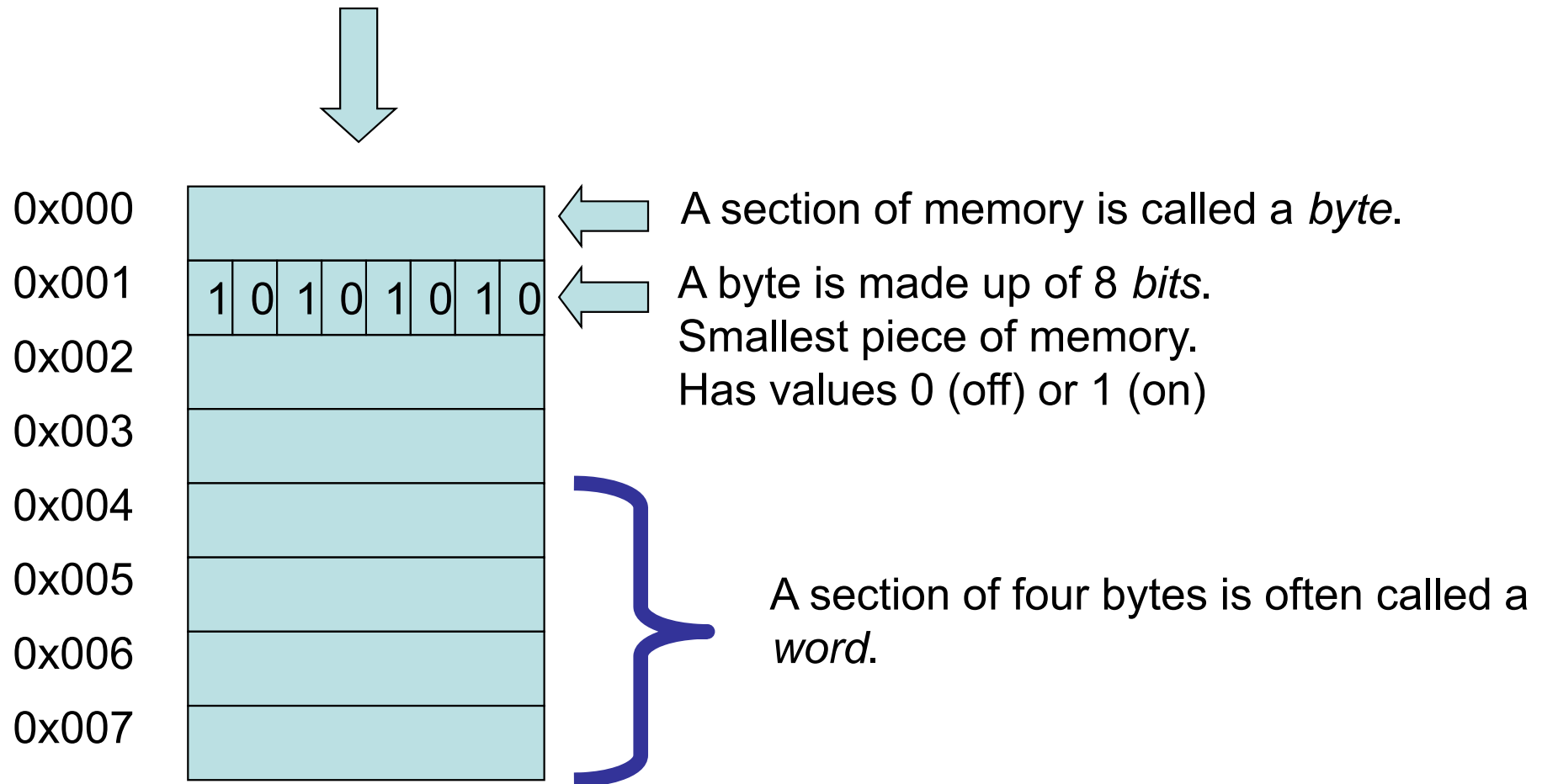


- 휘발성 메모리 사용 : 아래 경우에 데이터가 삭제
  - 프로그램 종료시
  - 컴퓨터 종료시



# main memory(cont.)

Main memory can be visualized as a column or row of cells.





# Hardware-CPU

- 프로그램을 실행하는 하드웨어
  - 중앙처리장치, Central Processing Unit
  - 제어장치와 산술논리연산장치, 레지스터로 구성
- 마이크로프로세서 (Microprocessor), 프로세서(Processor)

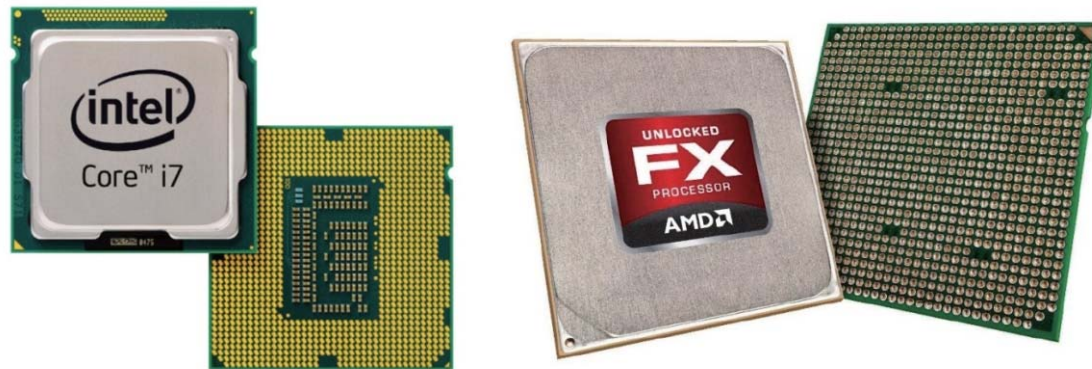
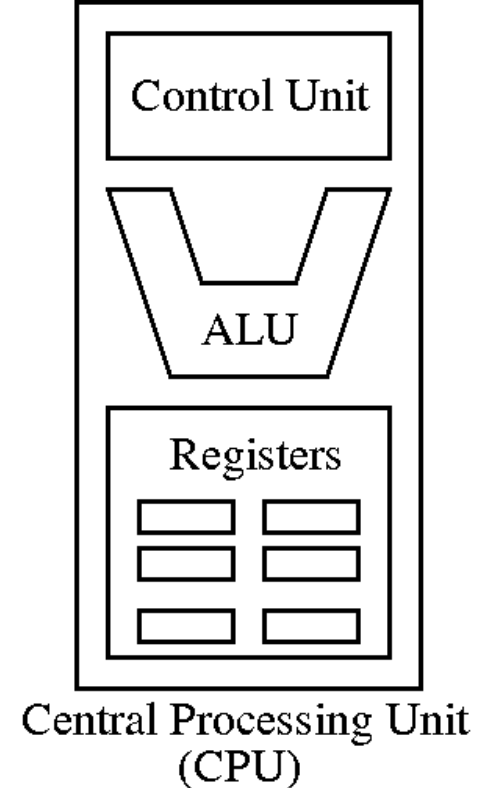


그림 2.3 인텔 사의 CPU와 AMD 사의 CPU

# Hardware-CPU(cont.)

- 산술논리연산장치(Arithmetic Logic Unit)
  - 실제로 프로그램이 실행되는 곳
  - 산술연산, 논리연산
- 제어 장치(Control Unit)
  - 명령어 해석, CPU의 두뇌
- 레지스터(Registers)
  - 임시적으로 데이터를 저장하기 위한 메모리
- 프로그램 실행 사이클
  - Fetch: get the next program instruction from main memory
  - Decode: interpret the instruction and generate a signal
  - Execute: route the signal to the appropriate component to perform an operation



# Software

- Operating System(OS)
  - 사용자(또는 사용자 프로그램)와 하드웨어 사이의 인터페이스
  - 제한된 시스템 자원(하드웨어)을 제어 및 관리
    - CPU 스케줄링, 메모리 할당
    - 하드웨어 장치를 최대한 효율적으로 작동시켜 컴퓨터 처리력을 높이기 함
  - 사용자가 컴퓨터와 상호 작용할 수 있는 방안을 제공
    - 시스템을 보다 편리하게 사용하고 질 좋은 서비스를 받게 함
    - 운영체제가 먼저 제대로 실행되어야만(부팅되어야만) 사용자가 사용할 수 있는 상태가 됨
  - Windows, Unix, Mac, Linux, iOS, android
- Application software
  - 사용자 작업에 특화된 환경 제공
  - 워드프로세서, 웹 브라우저, 스프레드시트, 게임...

# What we know:

- 프로그램은 실행시에 메인 메모리에 저장되고, CPU가 한 개의 명령어씩 처리한다.
- 메모리는 0 또는 1을 저장한다.
- 프로그램은 0 또는 1로 표현(기계어)된다.
  - 기계어: 컴퓨터가 직접 이해할 수 있는 유일한 언어

Q. 0 또는 1로 어떻게 프로그래밍하나?

A. 할 수는 있으나 아주 불편.

# 프로그래밍 언어들

Generation	Characteristics	Examples
1 <sup>st</sup>	Machine language	0110, 1100, 0000, ...
2 <sup>nd</sup>	Assembly language	ADD, SUB, MUL, ...
3 <sup>rd</sup>	High-level language	C, C++, <u>Java</u>
4 <sup>th</sup> / 5 <sup>th</sup>	Application-specific	LabVIEW, PHP, SQL
	Scripting languages	Perl, Python, Ruby, Unix Shell
	Mathematical	Mathematica, Simulink of MATLAB, R, ...

English:	<b>ADD</b>	contents of <b>ADDRESS1</b>	and contents of <b>ADDRESS2</b>	then store result in <b>ADDRESS3</b>
Machine language:	01011010	00000001	00000010	00000011
Assembly language:	add	a1,	a2,	a3
High-level language:	total = number1 + number2			

# 자바

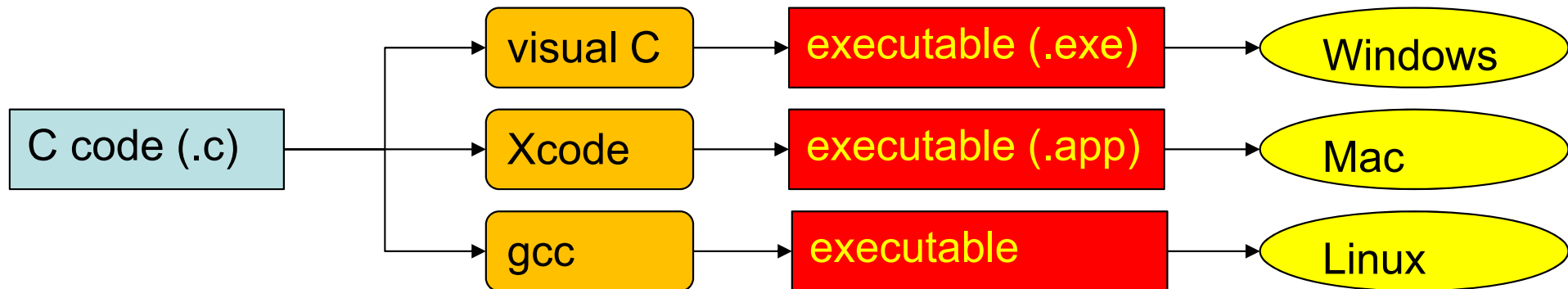
- 썬 마이크로시스템즈의 제임스 고슬링 (James Gosling)
- 그린 프로젝트 (1991)
  - TV, 냉장고.. 등의 가전제품에 탑재될 수 있는 프로그램 개발
  - 가전제품들이 네트워크로 연결되어 상호작용
  - 플랫폼에 독립적인 언어 개발하고자 함
  - C++ 언어로 개발⇒너무 복잡 ⇒ Oak 개발
- Oak 상표등록 어려움 ⇒ 언어 이름을 자바로 변경
- 인터넷 브라우저인 넷스케이프 내비게이터에 탑재
- 그 이후 다양한 애플리케이션, 핸드폰, 뽀빠, PDA, 스마트 폰에서 사용됨



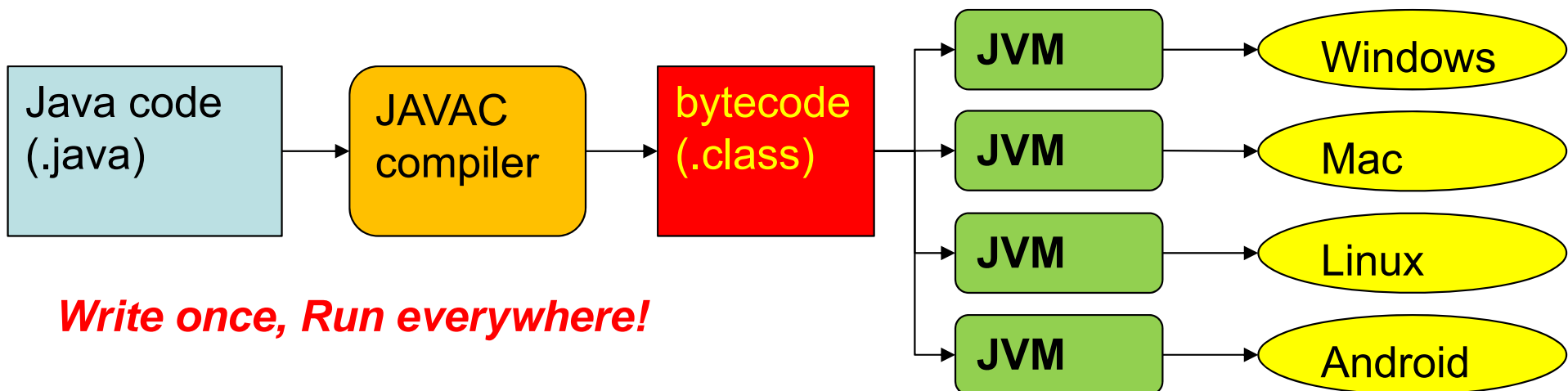
# 플랫폼에 독립적



Ordinary languages: platform dependent binaries



**Java: platform-independent bytecode, if platform-dependent JVM exists**



***Write once, Run everywhere!***

# .java

# 바이트 코드

```
public class Hello {  
    public static void main(String[] args) {  
        System.out.println("Hello");  
    }  
}
```

## .class

Compiled from "Hello.java"

```
public class Hello extends java.lang.Object {  
    public Hello();
```

Code:

0: aload\_0

1: invokespecial #1; // Method java/lang/Object."<init>":()V

4: return

```
public static void main(java.lang.String[]);
```

Code:

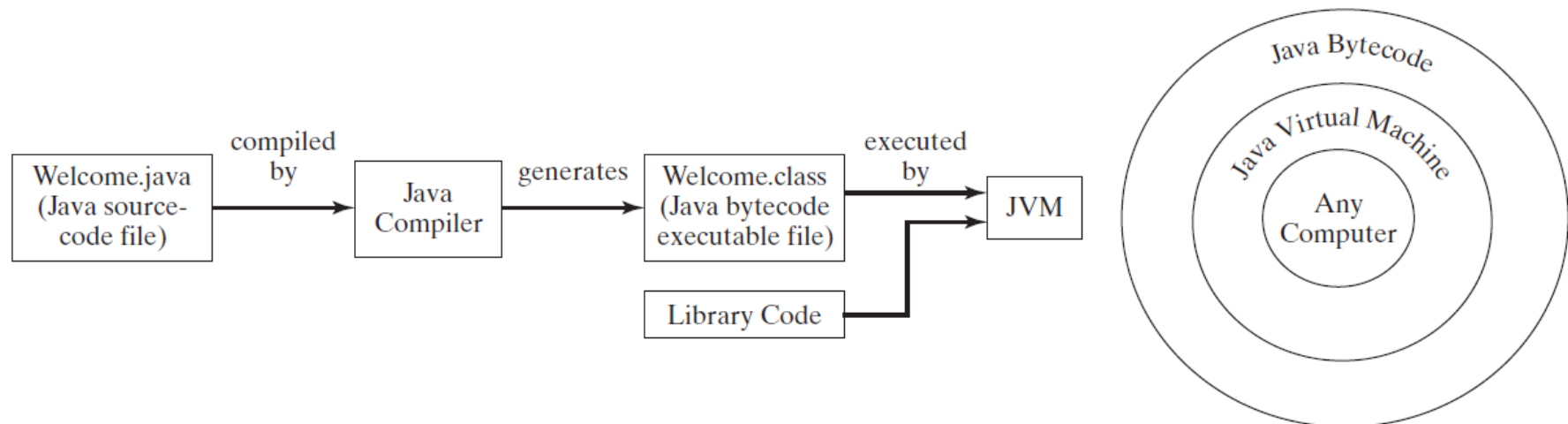
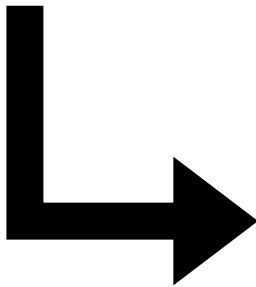
0: getstatic #2; // Field java/lang/System.out:Ljava/io/PrintStream;

3: ldc #3; // String Hello World!

5: invokevirtual #4; // Method java/io/PrintStream.println:  
(Ljava/lang/String;)V

8: return

**Similar to Assembly  
language, but  
machine-independent**





# 절차적 프로그래밍 vs. 객체지향 프로그래밍

- 절차적 프로그래밍
  - 순차적 진행 : do step A; do step B; do step C
  - 조건 : if X is true, do step A; otherwise do step B
  - 반복 : do step A while X is true
- 객체지향 프로그래밍
  - 클래스(Class) : 특정 목적을 가진 코드의 단위, 메서드들로 구성
  - 메서드 (method) : 특정 작업을 수행하는 코드 단위,
  - 명령어들로 구성

# 자바 기본 프로그래밍

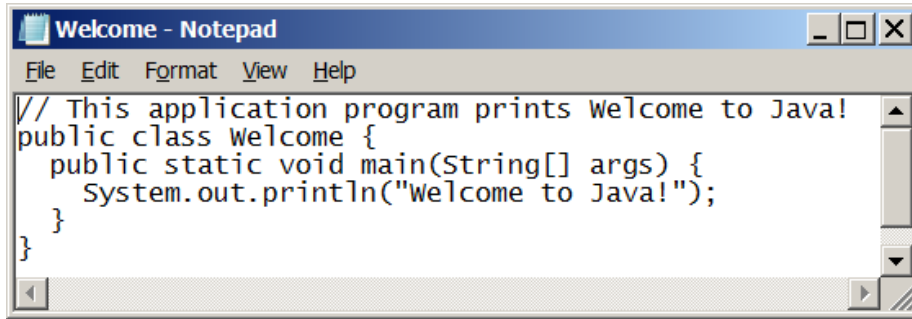
- Hello world 출력
- 개발환경 구축(별도 자료)

# 자바 개발 환경

- JDK 설치
  - java SE  
(Standard Edition)

- JRE (Java Runtime Environment)
  - 자바 실행 환경
  - JVM (java virtual machine), runtime libraries, etc.
- JDK (Java Development Kit)
  - 자바 언어를 이용하여 프로그램을 개발하기 위한 최소한의 환경
  - JRE + compiler + debugger, etc.
- Java IDE (integrated development environment)
  - 프로그램 개발에 필요한 컴파일러(compiler), 디버거(debugger), 링커(linker), 에디터(editor) 등을 통합적으로 제공하는 개발 환경
  - Popular Java IDEs
    - Eclipse, Android Studio, NetBeans, etc.

# 자바 코드 작성, 컴파일, 실행



```
// This application program prints Welcome to Java!
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

Source code (developed by the programmer)

```
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

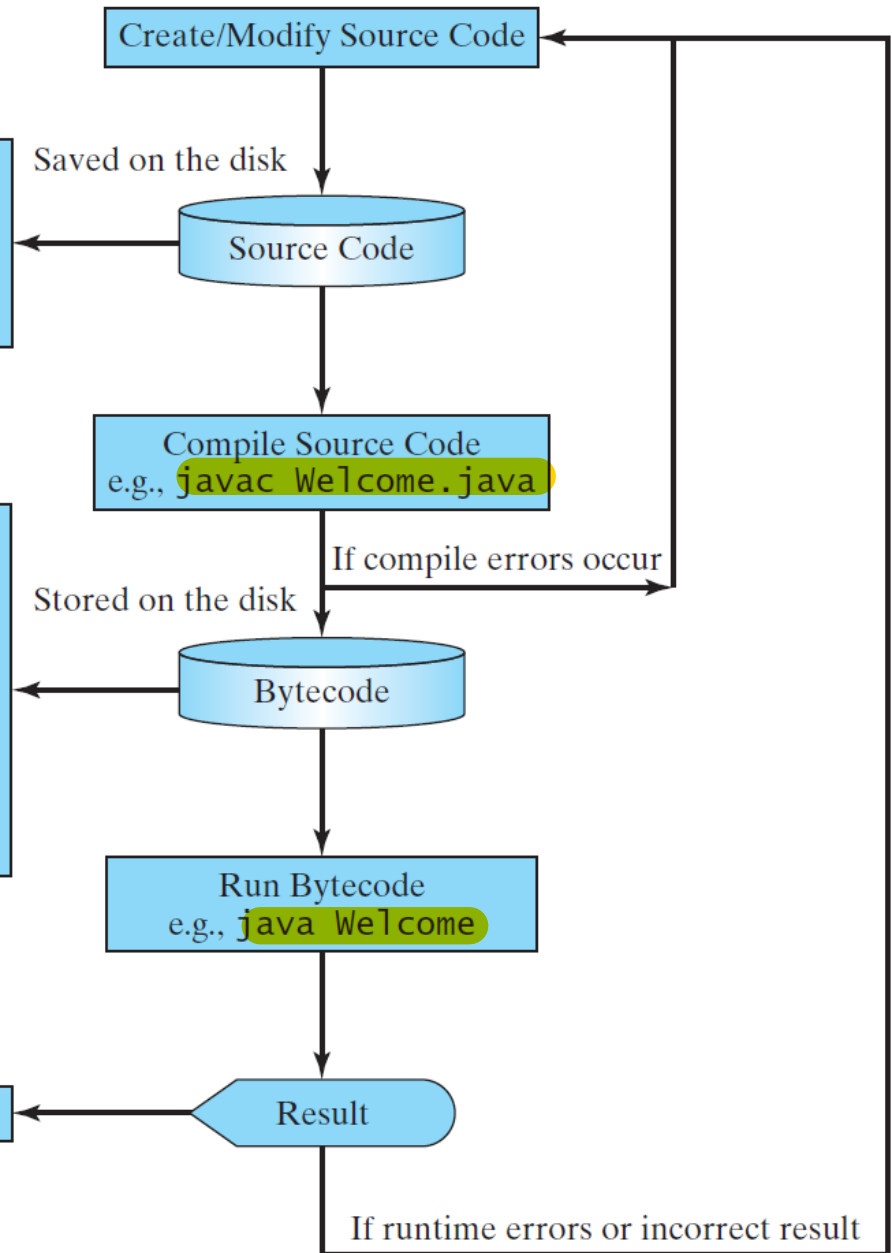
Bytecode (generated by the compiler for JVM  
to read and interpret)

```
...
Method Welcome()
  0 aload_0
  ...

Method void main(java.lang.String[])
  0 getstatic #2 ...
  3 ldc #3 <String "Welcome to Java!">
  5 invokevirtual #4 ...
  8 return
```

“Welcome to Java” is displayed on the console

Welcome to Java!



# Welcome to Java! 출력

- Class
  - 자바 프로그램은 적어도 하나의 클래스를 포함
    - 자바 프로그램의 기본 단위
    - 필드와 메소드로 구성
  - 관례상, 클래스명의 첫 글자는 대문자로
    - 당분간 파일명은 클래스명과 일치시킬 것(public 클래스인 경우 파일명과 클래스명은 동일)
- main method
  - CPU는 main() 메소드 내부의 코드를 실행함

```
// This program prints Welcome to Java!
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

# Welcome to Java! 출력

- Statement

- 프로그램 명령의 기본 단위
- 모든 문장은 ;으로 끝남

```
... main(...)  
{  
    statement;  
    statement;  
    statement;  
    ....  
}
```

- Blocks { }

- 명령어들을 그룹화 시켜줌

```
public class Test {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```

← Class block

← Method block

- """

- 문자열을 의미

# Welcome to Java! 출력

- 주석
  - 컴파일러가 무시하는 문장
  - //(한 라인), /\* ... \*/ (여러 라인)

```
/* John Smith, 2016001  
   COMP217099, Java Programming */  
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```

# 들여쓰기(Indentation)

- 클래스 정의에서 필드나 메소드의 첫 글자는 탭(tab)만큼 들여 쓰는 방식
  - 블록의 첫 문장은 메소드 헤더에서 탭만큼 다시 들여 써야 함
  - 프로그램 작성에 익숙하지 않은 초보자에게는 매우 중요한 코딩 방식
- 블록 스타일: end-of-line style for braces 추천

*Next-line style*

```
public class Test
{
    public static void main(String[] args)
    {
        System.out.println("Block Styles");
    }
}
```

*End-of-line style*

```
public class Test {
    public static void main(String[] args) {
        System.out.println("Block Styles");
    }
}
```



# 변수

## 메모리 공간

- Scanner 클래스를 사용해서 사용자의 입력 받기
- 변수의 자료형

Primitive data types (non-class)

Integer / Real / Boolean / Char / String

# 변수



- **변수(variable)**

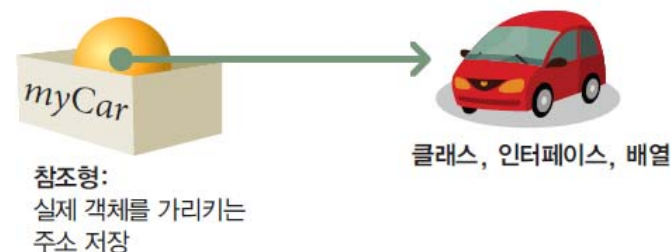
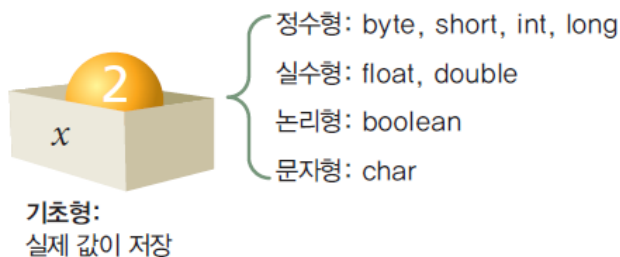
- 데이터 값들이 저장되는 메모리 공간
- 프로그램 실행 도중 저장된 값의 변경이 가능

- **자료형(data type)**

- 자료의 타입
- 물건을 정리하는 상자도 다양한 타입이 있듯이 자료를 저장하는 변수도 다양한 종류가 있다.



- 기초형과 참조형으로 나뉘어진다.



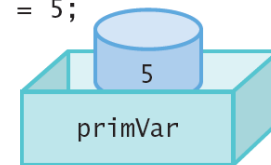
# 자바의 자료형

- 기본형
  - 변수의 저장 공간에 값 자체가 저장
- 참조형
  - 변수의 저장 공간에 참조 값이 저장

구분	분류	키워드
기본형	논리형	boolean
	문자형	char
	정수형	byte, short, int, long
	실수형	float, double
참조형	배열	int [], float [] 등 다양
	클래스	String, Date 등 다양
	인터페이스	Runnable, Enumeration 등 다양

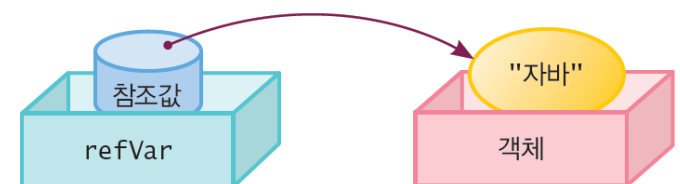
기본형

```
int primVar = 5;
```

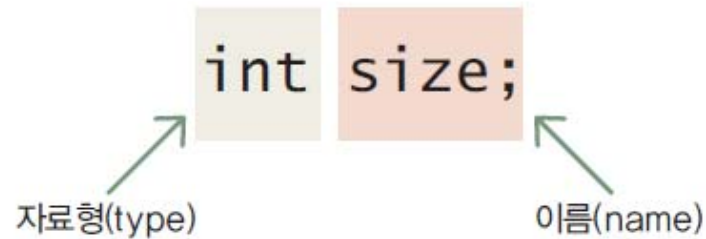


참조형

```
String refVar = new String("자바");
```

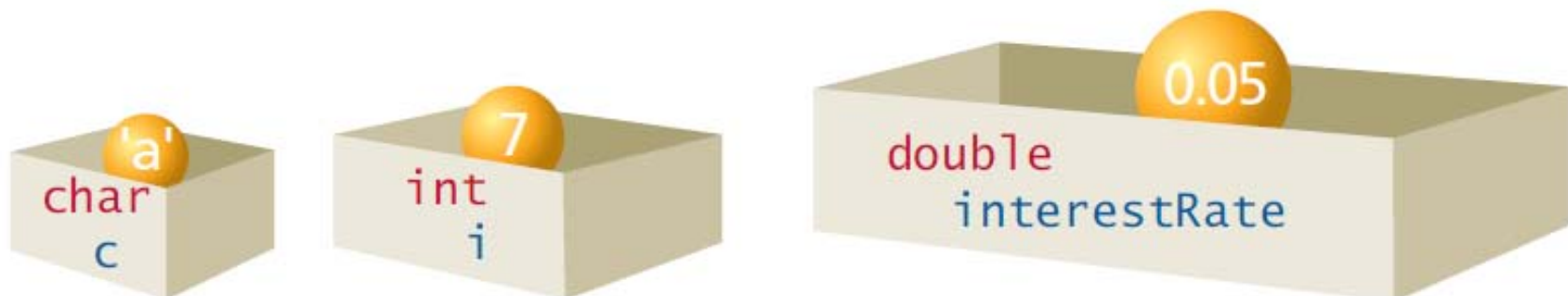


# 변수의 선언과 초기화



```
char c = 'a';           // allocate 2 byte memory location, and store 'a' in it
int i = 7;               // allocate 4 bytes memory location, and store 7 in it
double interestRate = 0.05; // allocate 8 bytes memory location, and store 0.05 in it
```

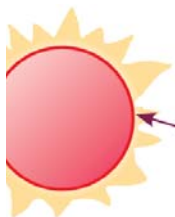
- `int index, total = 0;` //하나의 문장에서 변수를 여러 개 선언할 수도 있다.
- `int index, char c;` //에러!! 다른 타입을 한 문장에 정의할 수 없다.



# 자료형의 크기

- 정수와 실수를 표현하는 자료형이 다양한 이유
  - 그 표현 범위가 다르기 때문

분류	키워드	크기	상대적 크기 비교	최소 ~ 최대	지수형태 범위
논리형	boolean	1바이트	■	false, true	
문자형	char	2바이트	■ ■	\u0000 ~ \uffff, [0 ~ 65,535]	$0 \sim 2^{16}-1$
정수형	byte	1바이트	■	-128 ~ 127	$-2^7 \sim 2^7-1$
	short	2바이트	■ ■	-32,768 ~ 32,767	$-2^{15} \sim 2^{15}-1$
	int	4바이트	■ ■ ■ ■	-2,147,483,648 ~ 2,147,483,647	$-2^{31} \sim 2^{31}-1$
	long	8바이트	■ ■ ■ ■ ■ ■ ■ ■	$-2^{63} \sim 2^{63}-1$	$-2^{63} \sim 2^{63}-1$
실수형	float	4바이트	■ ■ ■ ■	(+, -)1.4E-45 ~ 3.4028235E38	
	double	8바이트	■ ■ ■ ■ ■ ■ ■ ■	(+, -)4.9E-324 ~ 1.7976931348623157E308	

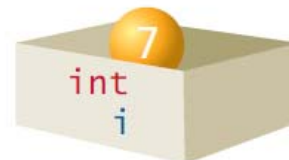


태양

태양과 지구 사이 거리 1500억 m는  
21억을 초과하므로 자료형 long 이용

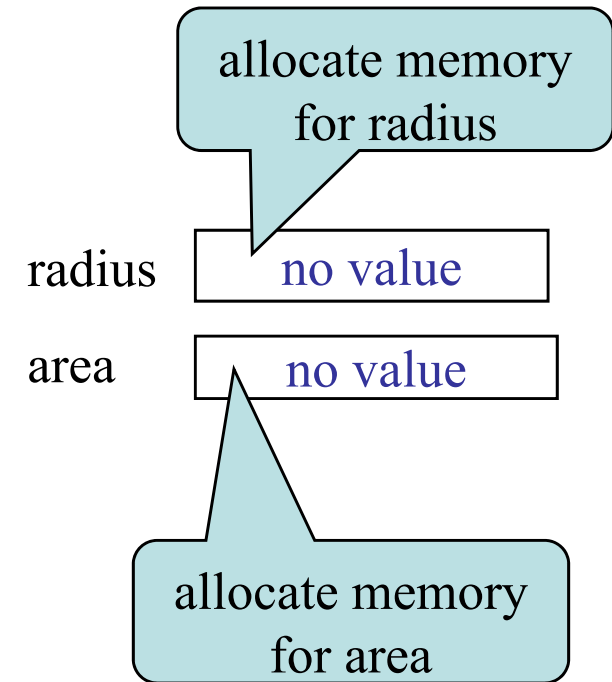


만일 km로 저장한다면 1.5억 km이므로  
자료형 int에도 저장이 가능함



# Trace a Program Execution

```
public class ComputeArea {  
    /** Main method */  
    public static void main(String[] args) {  
        double radius;  
        double area;  
  
        // Assign a radius  
        radius = 20;  
  
        // Compute area  
        area = radius * radius * 3.14159;  
  
        // Display results  
        System.out.println("The area for the circle of  
        radius " +  
        radius + " is " + area);  
    }  
}
```



# Trace a Program Execution

```
public class ComputeArea {  
    /** Main method */  
    public static void main(String[] args) {  
        double radius;  
        double area;  
  
        // Assign a radius  
        radius = 20;  
  
        // Compute area  
        area = radius * radius * 3.14159;  
  
        // Display results  
        System.out.println("The area for the circle of  
        radius " +  
        radius + " is " + area);  
    }  
}
```

radius

assign 20 to radius

20

area

1256.636

compute area and assign it  
to variable area

# Trace a Program Execution

```
public class ComputeArea {  
    /** Main method */  
    public static void main(String[] args) {  
        double radius;  
        double area;  
  
        // Assign a radius  
        radius = 20;  
  
        // Compute area  
        area = radius * radius * 3.14159;  
  
        // Display results  
        System.out.println("The area for the circle of  
        radius " +  
        radius + " is " + area);  
    }  
}
```

memory

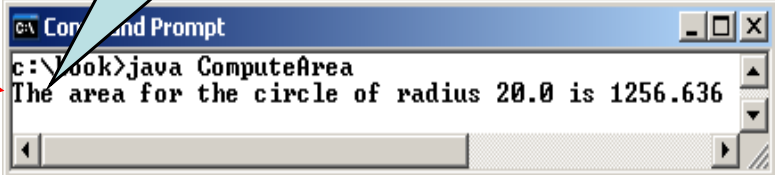
radius

20

area

1256.636

print a message to the  
console



```
C:\book>java ComputeArea  
The area for the circle of radius 20.0 is 1256.636
```



# 식별자

## (Class, Variable, Method 등의 이름)

- 식별자 구성 문자

- 식별자는 대소문자 알파벳, 숫자(0~9), 밑줄(\_), 달러 기호(\$)로 구성
- 규칙

- ① 키워드는 식별자로 이용할 수 없다

- 즉 키워드 byte, case 등은 식별자가 될 수 없다.

- ② 식별자의 첫 문자로 숫자가 나올 수 없다.

- 즉 77fortran, 2020year 등은 식별자가 될 수 없다.

- ③ 식별자는 대소문자를 구별

- 변수 Count, count, COUNT는 모두 다른 변수로 취급
- 중간에 공백space 문자가 들어갈 수 없다.

- ④ 자바는 유니코드를 지원하므로 식별자로 한글을 이용 가능

- 실무 프로그램에서 한글 사용은 권장하지 않는다

# 키워드와 식별자 관례

- 프로그래밍 언어에서 문법적으로 의미 있는 단어로 사용하기 위해 미리 정의해 놓은 단어
  - 키워드는 예약어(reserved word)라고도 함

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

Cases	Conventions	Examples
복합단어로 구성된 클래스명	각 단어의 첫글자는 대문자	StaffMember ItemProducer
변수, 메서드 명	소문자로 시작, 2번째 단어의 첫글자는 대문자	width, payRate, acctNumber, genMonthDays(), fillRect()
상수	모든 문자를 대문자로	MAX_NUMBER

# 변수 이름의 예

```
public class Identifiers {  
    public static void main (String args[] ) {  
        // 유효한 식별자들  
        int speed;           // 일반적인 단어  
        long earchPopulation; // 단어 연결  
        int _count;          // '_'로 시작  
        long $value;          // '$'로 시작  
        long Henry8;          // 숫자로 끝남  
        long a2c4e6f8i;       // 숫자와 문자 조합 (숫자로 시작하지 않음.)  
  
        // 유효하지 않은 식별자들  
        int 1stPrizeMoney;  
        double class;  
        double the#ofComputer;  
        double abc@def.com;  
        double knu.ac.kr;  
    }  
}
```

# 정수형-Example 1

```
1 public class Integer_test{
2     public static void main(String args []){
3         int a = 14, b = 15, c = 16; //10진수 : 14,15,16과 같이 소스 코드에 쓰여
        //있는 값을 상수 또는 리터럴(literal)이라고 함
4         int d = 014, e = 015, f = 016; //8진수 : 상수 앞에 0을 붙임
5         int g = 0x14, h = 0x15, i = 0x16; //16진수 : 상수 앞에 0x를 붙임
6         int j = 0b10010, k = 0b10, l = 0b1111; //2진수 : JDK 7부터 가능
7
8         System.out.println("a = " + a + ", b = " + b + ", c = " + c); //십진수로
        //표현
9         System.out.printf("a = %d, %#o, %#x\n", a, a, a); //a를 10진수, 8진수,
        //16진수 형식으로 표현
10        System.out.printf("a = %1$d, %1$o, %1$x\n", a); //위의 지시랑 같은
        //의미, 해당 인덱스의 아그먼트에서 읽어올 것
11        System.out.println("d = " + d + ", e = " + e + ", f = " + f);
12        System.out.println("g = " + g + ", h = " + h + ", i = " + i);
13        System.out.println("j = " + j + ", k = " + k + ", l = " + l);
14
15        byte number = 300;
16        System.out.println("number = " + number);
17        int e2 = 018, f2 = 019;
18        System.out.println("e2 = " + e2 + ", f2 = " + f2);
19    }
20 }
```

a = 14, b = 15, c = 16

a = 14, 010, 0x0e  
d = 12, e = 13, f = 14  
g = 20, h = 21, i = 22  
j = 18, k = 2, l = 15

# 정수형-Example 2

```
public class 빛이가는거리{  
    public static void main(String args []){  
        long lightspeed;  
        long distance;        // 8바이트(64 비트) 정수 저장 가능  
        double dist_f;        // 8바이트(64 비트) 실수 저장 가능  
  
        lightspeed = 300_000;    // 300,000 km/s  
        distance = lightspeed * 365L * 24 * 60 * 60;  
        dist_f = (double) distance; // long 타입을 double 타입으로 변환  
  
        System.out.println("빛이 1년 동안 가는 거리 : " + distance + " km.");  
        System.out.printf("빛이 1년 동안 가는 거리 : %e km.\n", dist_f);  
    }  
}
```

- 자바에서는 아스키코드가 아니라 유니코드를 사용하여 한글 지원
  - 클래스, 변수 명에 한글 사용 가능
  - 실무에서는 사용 권장하지 않음

# 실수형-Example

```
public class Real_test {
    public static void main (String args[] ) {
        //float temp_f1 = 25.6;      // 에러!! 25.6은 기본적으로 double 타입
        float temp_f2 = 25.6f;      // OK or 25.6f
        double temp_d1 = 37.5;      // OK: 37.5 is double (default)
        double temp_d2 = 37.5F;     // OK, why?
        double _underbar = 123_456_789.0; // 가독성을 높이기 위해 실수형에도 _가능 (JDK 7부터)
        System.out.println("123_456_789.0 = " + _underbar);

        final double PI = 3.141592; // 기호 상수, 숫자보다 이해하기 쉽고 값의 변경이 용이
        //PI = PI * 2;              // 에러!! 상수의 값은 바뀔 수 없음

        System.out.println("500.0 / 0 = " + 500.0 / 0); // 무한대
        System.out.println("-500.0 / 0 = " + -500.0 / 0); // 음의 무한대
        System.out.println("0 / 0 = " + 0 / 0); // 0을 0으로 나눔
    }
}
```

- 실수형 자료는 double을 사용하는 것이 바람직

## 실행결과

```
123_456_789.0 = 1.23456789E8
500.0 / 0 = Infinity
-500.0 / 0 = -Infinity
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at Real_test.main(Real_test.java:15)
```

# double vs. float

Floating-point real numbers: 5.0f or 5.0F

Double-precision numbers: 5.0d or 5.0D (default)

The double type values are more accurate than the float type values. For example,

```
System.out.println("1.0 / 3.0 is " + 1.0 / 3.0);
```

displays **1.0 / 3.0 is 0.3333333333333333**



16 digits

```
System.out.println("1.0F / 3.0F is " + 1.0F / 3.0F);
```

displays **1.0F / 3.0F is 0.33333334**



7 digits

# 논리형-Example

```
public class Boolean_test {  
    public static void main (String args[] ) {  
        boolean b;           //c/c++과는 달리 0 or 다른 숫자로 표현 불가능  
                               //true or false 두가지 값만을 가짐.  
  
        b = true;  
        System.out.println("b = " + b);  
  
        b = (1>2);           //결과 값은 false  
        System.out.println("b = " + b);  
    }  
}
```

```
/* 실행결과  
javac Boolean_test.java  
java Boolean_test  
b = true  
b = false  
*/
```



# 문자형과 문자열

```
public class Char_test {
    public static void main(String[] args) {
        char    c1,
               c2,
               c3;

        c1 = 'a';
        System.out.println("c1 : "+c1); //출력하고자하는 것을 + 를 통해 연결
        c2 = '가';
        System.out.println("c2 : "+c2);
        c3 = '\uac00';                // 16진수 숫자 4개로 유니코드 표현 가능
        System.out.println("c3 : "+c3);

        char    c4=' , ';
        String s1 = "Hello";
        String s2 = "Java";
        System.out.println(s1);
        System.out.println(s2);
        System.out.println(s1+c4+" "+s2);
    }
}
```

## 실행결과

```
c1 : a
c2 : 가
c3 : 가
Hello
Java
Hello, Java
```

- 문자열은 String 클래스로 제공된다.
- 문자열 변수를 선언하려면 String 타입을 사용한다.

# Constants (1): literals

- A constant is a number whose value do not change during program execution
  - Two ways: named constants and literals
- Literal
  - Numeric literals are numbers
    - Example: 7 or 3.14159
  - Character literals are written in single quotes
    - Example: 'A' or '&'
  - String literals are enclosed in double quotes
    - Example: "Hello"
    - Strings are concatenated by plus operator (+)

# Constants (2): named constants

- Named constants
  - A named constant is a memory location whose content cannot change during execution
  - However, a variable is a memory location whose content may change during program execution
- Use the keyword '**final**'
  - By convention, named constants are often uppercase, but not always
    - **final double TAX\_RATE = .15;**
    - **final double PI = 3.14159;**

# Receiving Inputs from Keyboard

# 키보드로 입력받기: Scanner 클래스

- Scanner:
  - 자바 라이브러리 중 하나: java.util 패키지에 포함
    - 패키지: 관련있는 자바 클래스가 모여있는 폴더
  - 키보드나 파일로부터 데이터를 입력 받을 때 사용
- `import java.util.Scanner;`
  - 컴파일러에게 이 프로그램이 Scanner클래스를 사용함을 알려줌
- `Scanner s = new Scanner(System.in);`
  - Scanner 클래스의 객체 생성
  - System.in은 표준 입력을 의미
  - 사용자로부터 입력을 받으려면 반드시 스캐너 클래스의 객체부터 생성할 것

# Scanner 클래스

- Scanner 클래스는 커맨드 윈도우에서 데이터를 입력받는데 사용

```
Scanner input = new Scanner(System.in);  
int value = input.nextInt();
```

Method	Description
<code>nextByte()</code>	reads an integer of the <code>byte</code> type.
<code>nextShort()</code>	reads an integer of the <code>short</code> type.
<code>nextInt()</code>	reads an integer of the <code>int</code> type.
<code>nextLong()</code>	reads an integer of the <code>long</code> type.
<code>nextFloat()</code>	reads a number of the <code>float</code> type.
<code>nextDouble()</code>	reads a number of the <code>double</code> type.

# 실습

- 10년동안의 월급을 모두 저금할 경우, 얼마나 모을 수 있는지 계산하는 프로그램을 작성하시오.
  - 사용자로부터 월급을 입력 받음
  - 저금액 계산 후 출력
  - 필요한 변수: 월급
- 사용자로부터 원의 반지름을 입력 받아 넓이를 계산하는 프로그램을 작성하시오.
  - 원의 반지름 입력받기
  - 원 넓이 계산
  - 출력하기
  - 필요한 변수: 원의 반지름
- 직사각형의 가로와 세로를 입력 받아 넓이와 둘레를 계산하는 프로그램을 작성하시오.

# 중간 점검 문제 1

- ① 자바 프로그램의 개발 단위는 \_\_\_\_\_이다.
- ② \_\_\_\_\_은 프로그램에 대한 설명이다.
- ③ 입력을 받아서 작업을 수행하고 결과를 내보내는 작은 기계로 생각할 수 있는 것은 \_\_\_\_\_이다.
- ④ 모든 자바 소스 파일의 확장자는 \_\_\_\_\_이다.
- ⑤ 프로그램에서 데이터를 저장하는 공간은 \_\_\_\_\_이다.
- ⑥ 변수에 값을 저장하는 연산을 \_\_\_\_\_이라고 한다.
- ⑦ 실행 도중에 값이 변하지 않는 수를 \_\_\_\_\_이라고 한다.
- ⑧ 사용자로부터 값을 입력받으려면 \_\_\_\_\_클래스를 사용하는 것이 편리하다



## 중간 점검 문제 2

1. 변수가 36에서 5000정도의 값을 저장하여야 하다면 어떤 자료형이 최적인가?
2. 변수가 -3000에서 +3000까지의 값을 저장하여야 하다면 어떤 자료형이 최적인가?
3. 0.025를 지수 표기법으로 표기하여 보라.
4. 어떤 리터럴(상수)이 더 많은 메모리 공간을 차지하는가?  
28.9      28.9F
5. boolean 자료형이 가질 수 있는 값을 전부 쓰시오.
6. 변수에 새로운 값이 대입되면 기존의 값은 어떻게 되는가?
7. days와 Days는 동일한 변수인가 아닌가?
8. 다음 중에서 올바르지 않은 변수이름은?  
x,    8items,    march09,    sales\_report,    theProfit2009,    #ofPlayer