

Java Programming

Introduction

소개

Computer Programming

: 간단한 이해

- ▶ 컴퓨터가 이해하는 언어는 1과 0으로 구성된 코드 (Machine Language: 기계어)
- ▶ 사람은 컴퓨터의 언어를 이해할 수 없고, 컴퓨터는 사람의 언어를 이해할 수 없다
- ▶ 사람의 언어와 기계 언어의 중간정도의 메타 언어가 필요
-> 프로그래밍 언어
- ▶ 저수준 언어 vs 고수준 언어
 - ▶ 성능상의 분류가 아님
 - ▶ 인간이 이해할 수 있는 언어 : 고수준 언어
 - ▶ 기계어에 가까운 언어 : 저수준 언어

High-level Language

```
temp = v[k];  
v[k] = v[k+1];  
v[k+1] = temp;
```

```
TEMP = V(K)  
V(K) = V(K+1)  
V(K+1) = TEMP
```

C/Java Compiler

Fortran Compiler

Assembly Language

```
lw $t0, 0($s2)  
lw $t1, 4($s2)  
sw $t1, 0($s2)  
sw $t0, 4($s2)
```

Machine Language

```
0000 1001 1100 0110 1010 1111 0101 1000  
1010 1111 0101 1000 0000 1001 1100 0110  
1100 0110 1010 1111 0101 1000 0000 1001  
0101 1000 0000 1001 1100 0110 1010 1111
```

MIPS Assembler

해석 방법에 따른 언어 분류

: Compiler vs Interpreter

- ▶ 컴파일러: 프로그래밍 언어를 미리 기계어로 바꾸어 두고 실행
-> 번역에 가까움



Figure: Compiler

- ▶ 인터프리터: 프로그래밍의 소스코드를 읽고 그 즉시 기계에게 실행을 요청
-> 통역에 가까움



Figure: Interpreter

Java Language

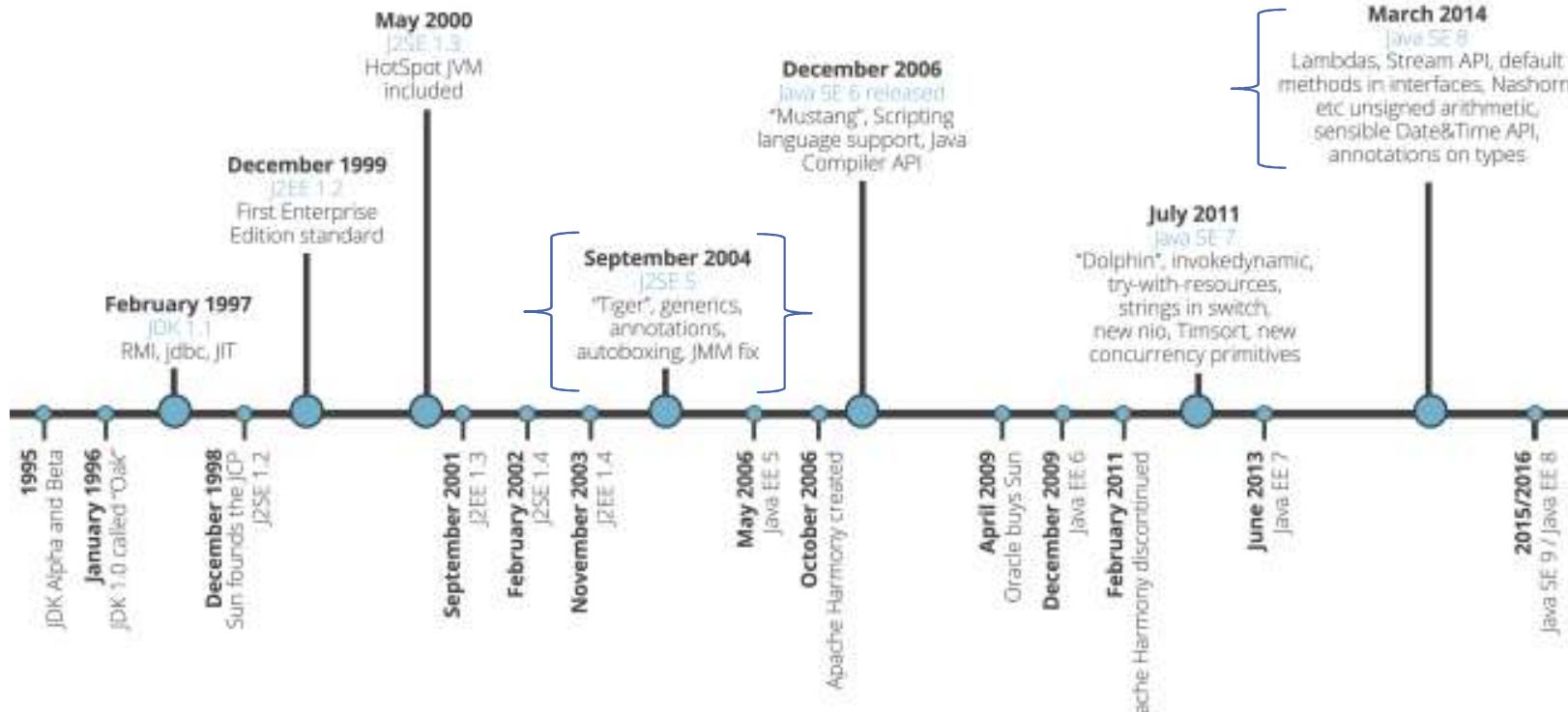
: Birth

- ▶ 자바 언어의 탄생
 - ▶ 1991년 미국 Sun Microsystems 사 제임스 고슬링(James Gosling)이 주도한 Oak 언어에서 출발
 - ▶ 초기에는 가전제품에서 사용할 목적으로 고안
 - ▶ 1995년, Sun World에서 공식 발표
 - ▶ 1996년, JDK 1.0 발표
 - ▶ 2009년, Oracle이 Sun Microsystems를 인수, 현재에 이름
 - ▶ 2017년 현재 9이 가장 최신 버전, 8을 가장 일반적으로 사용
- ▶ Write Once, Run Everywhere
 - ▶ 1996년(발표시점) ~ 1999년까지는 윈도 프로그래밍이 주류인 시장에서 열세
 - ▶ 1990년대 후반, 월드와이드웹(WWW)이 활성화되면서 웹 애플리케이션 구축 언어로 급부상
 - ▶ 다양한 장비와 환경에서 실행되는 애플리케이션을 개발하는 언어로 자리매김



Java Language

: Brief History



<https://zeroturnaround.com/rebellabs/a-short-history-of-nearly-everything-java/>

Java Language

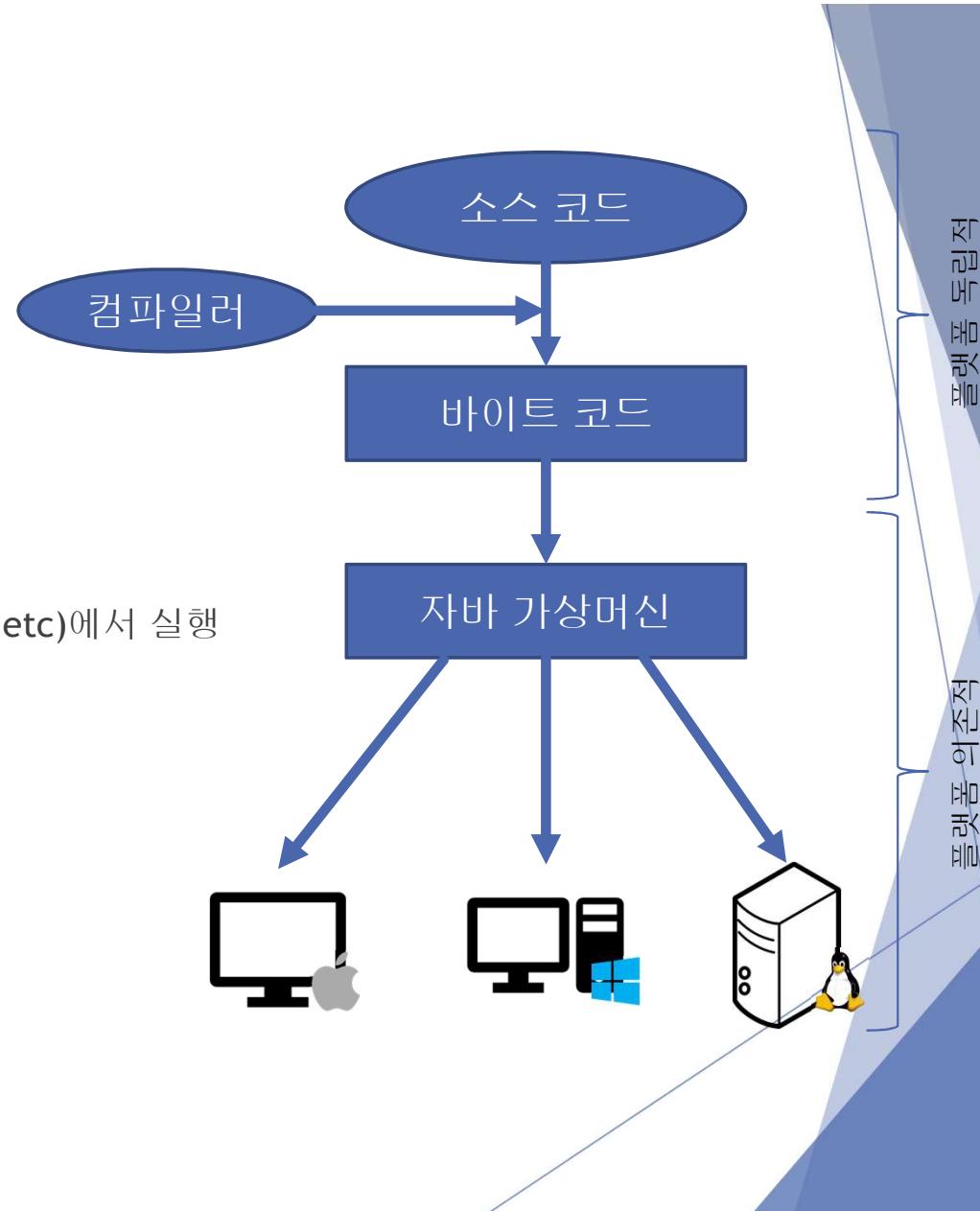
: Keywords

- ▶ Programming Language
- ▶ For Client-Server Programs (Both)
- ▶ Runs on all platforms
 - ▶ Via Java Virtual Machine (JVM)
- ▶ Syntax: derived from C/C++
- ▶ By Sun Microsystems (1991~1995)
- ▶ Object-Oriented
- ▶ Statically Typed

Java 특징 (1)

: 높은 이식성

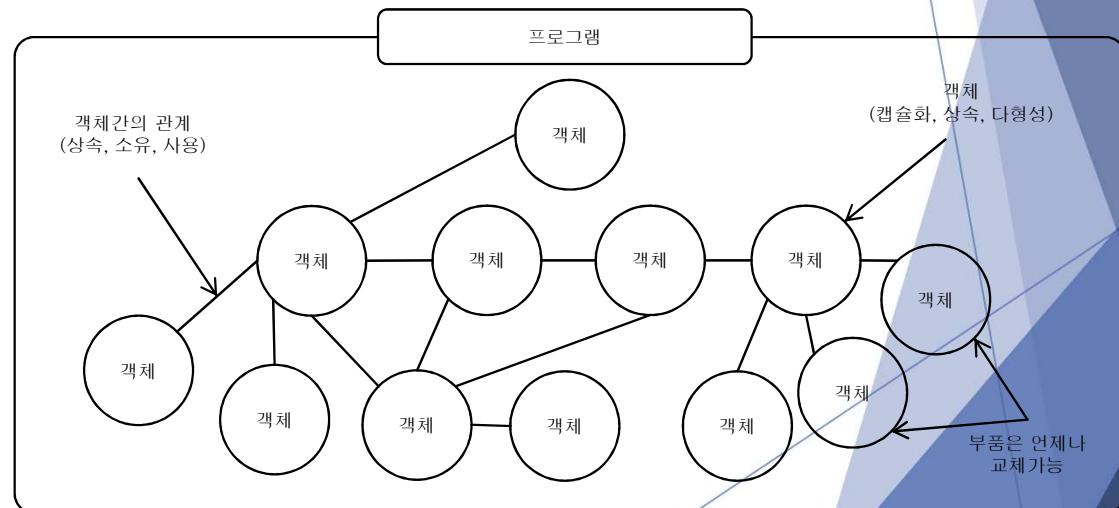
- ▶ Write Once, Run Everywhere
- ▶ JRE (Java Runtime Environment)
 - ▶ 소스 수정, 재컴파일 없이
 - ▶ 다양한 환경(Windows, Linux, Mac, etc)에서 실행



Java 특징 (2)

: 객체지향

- ▶ 100% 객체 지향 언어
- ▶ 객체지향 개발 : 서로 협력하는 다수의 객체들을 조합하는 개발 방식
 - ▶ 부품 - 완제품의 관계
 - ▶ 각 객체는 언제든 교체 가능
 - ▶ 현대 프로그래밍 패러다임 중 하나
- ▶ 캡슐화, 상속, 다형성 지원



Java 특징 (3)

: 단순한 문법과 개발 편의성

- ▶ C/C++ 언어의 구문과 C++ 객체지향에서 영향
 - ▶ 이해하기 어려운 포인터, 다중상속 등은 제외시킴
- ▶ 메모리 관리를 자바가 직접 관리
 - ▶ 개발자가 직접 메모리에 접근할 수 없도록 설계
 - ▶ Garbage Collector가 사용하지 않는 메모리를 제거
- ▶ Dynamic Language, 함수형 프로그래밍의 유용한 요소를 적극적으로 수용
 - ▶ 대용량 데이터의 병렬 처리, 이벤트 지향 프로그래밍 등에 유용 (Lambda 함수, NIO 등)
 - ▶ Java 8 이상부터 지원

Java 특징 (4)

: 그 외의 특징들

- ▶ 동적 로딩 (Dynamic Loading)
 - ▶ 필요한 시점에 객체를 동적 로딩
 - ▶ 변경이 필요한 부분만 수정하면 되므로 유지보수에 용이
- ▶ 풍부한 오픈 소스 라이브러리와 활성화된 개발자 생태계
- ▶ 범용 개발 언어
 - ▶ 다양한 운영체제, 다양한 디바이스에 대응하는 애플리케이션 개발 가능

Java 언어의 핵심 철학

: 5대 핵심 목표

- ▶ 객체 지향 방법론을 사용해야 한다
- ▶ 같은 프로그램(바이트 코드)이 여러 운영체제(마이크로프로세서)에서 실행될 수 있어야 한다
- ▶ 컴퓨터 네트워크 접근 기능이 기본으로 탑재되어 있어야 한다
- ▶ 원격 코드를 안전하게 실행할 수 있어야 한다
- ▶ 다른 객체 지향 언어들의 좋은 부분만 가지고 와서 사용하기 편해야 한다

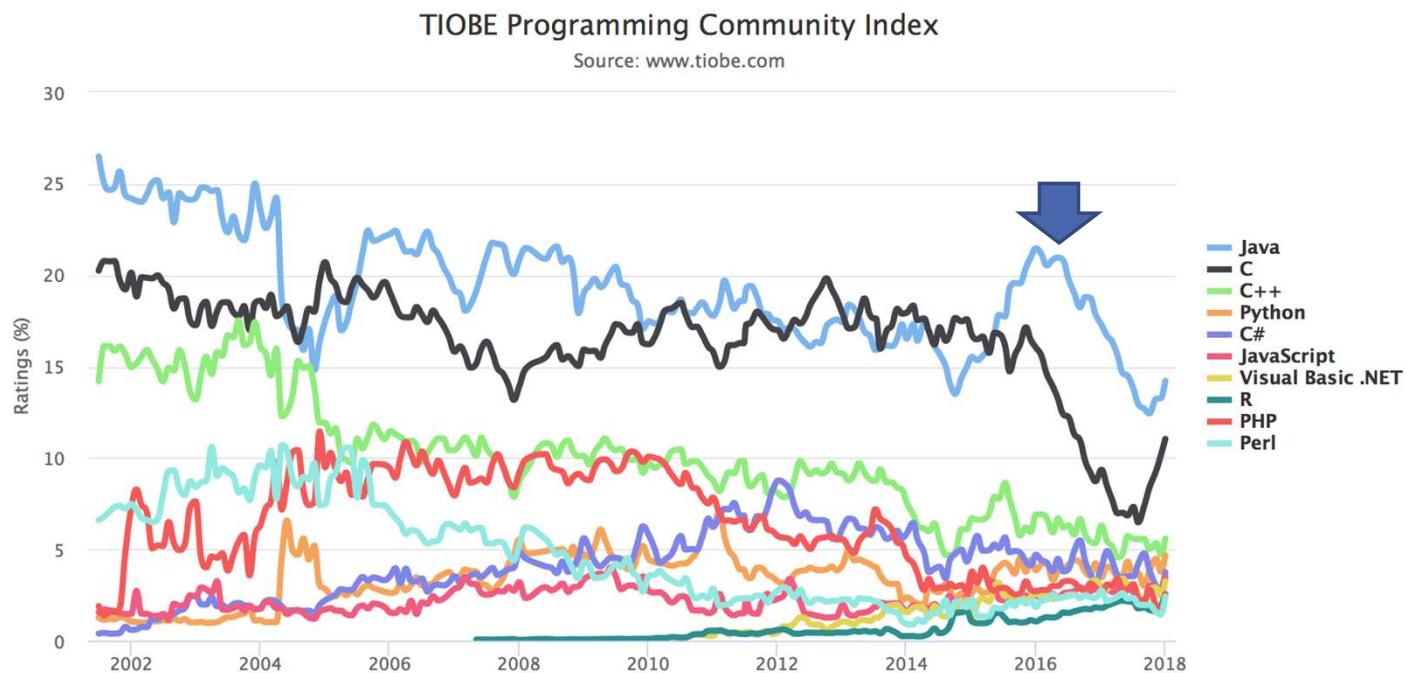
Java 언어의 현재와 미래

: TIOBE Index: 2018 December

Dec 2018	Dec 2017	Change	Programming Language	Ratings	Change
1	1		Java	15.932%	+2.66%
2	2		C	14.282%	+4.12%
3	4	▲	Python	8.376%	+4.60%
4	3	▼	C++	7.562%	+2.84%
5	7	▲	Visual Basic .NET	7.127%	+4.66%
6	5	▼	C#	3.455%	+0.63%
7	6	▼	JavaScript	3.063%	+0.59%
8	9	▲	PHP	2.442%	+0.85%
9	-	▲	SQL	2.184%	+2.18%
10	12	▲	Objective-C	1.477%	-0.02%
11	16	▲	Delphi/Object Pascal	1.396%	+0.00%
12	13	▲	Assembly language	1.371%	-0.10%
13	10	▼	MATLAB	1.283%	-0.29%
14	11	▼	Swift	1.220%	-0.35%
15	17	▲	Go	1.189%	-0.20%
16	8	▼	R	1.111%	-0.80%
17	15	▼	Ruby	1.109%	-0.32%
18	14	▼	Perl	1.013%	-0.42%
19	20	▲	Visual Basic	0.979%	-0.37%
20	19	▼	PL/SQL	0.844%	-0.52%

Is Java Dead?

: 자바의 위기



Is Java Dead?

: 위기의 원인과 모던 자바의 역습

▶ Java의 위기

- ▶ 다양해진 플랫폼과 다양한 언어의 등장
- ▶ 경쟁 언어들에 비해 더딘 프로그래밍 패러다임 도입 및 언어적 개선
- ▶ JVM 언어의 약진
: Scala, Groovy, Kotlin 등 자바 언어보다 쉬우면서 최신 프로그래밍 패러다임을 도입하면서도 JVM의 성능을 그대로 활용할 수 있는 언어들의 도래

▶ 모던 Java의 역습

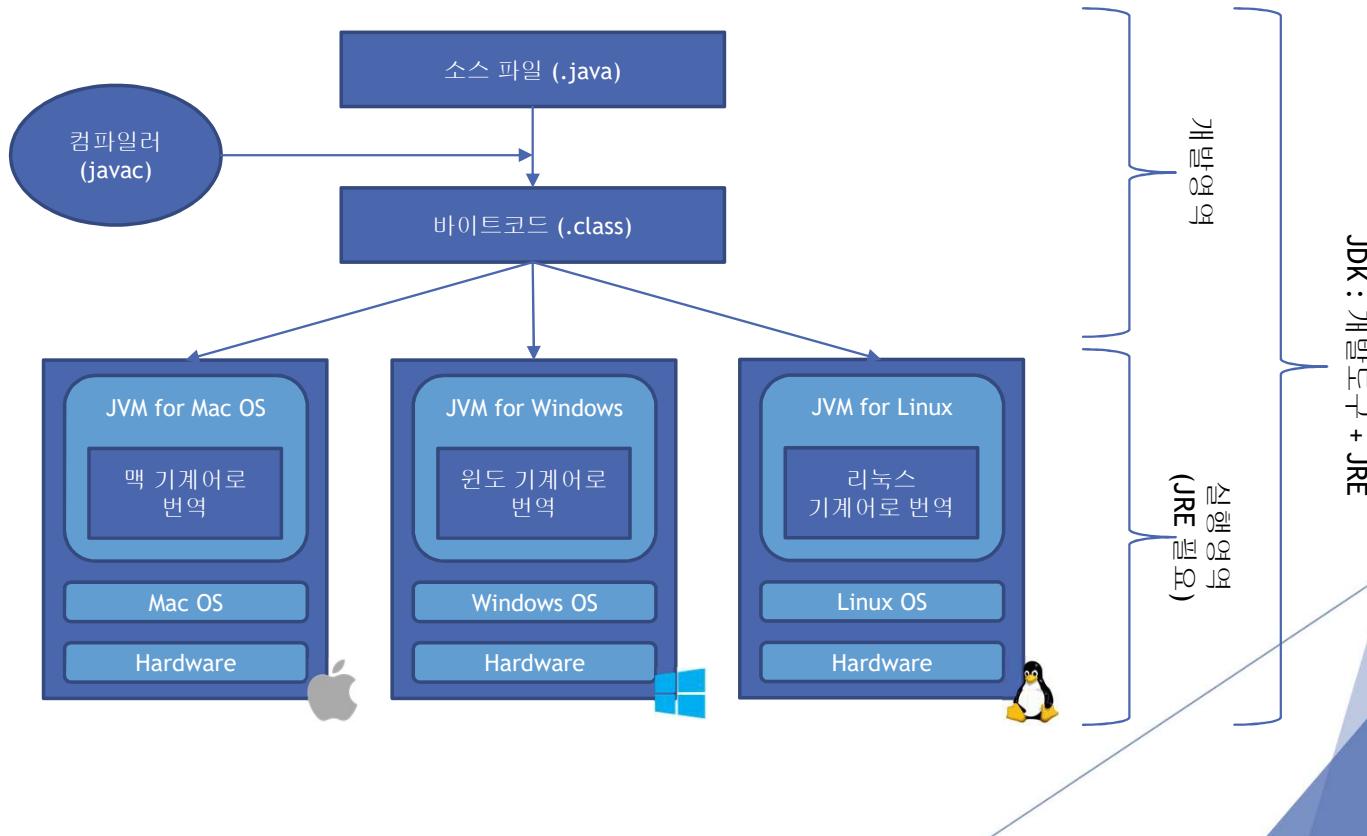
- ▶ Java 8, 9을 거치며, 언어적 개선과 최신 프로그래밍 패러다임을 적극적으로 수용
- ▶ 2018년 Java 11 LTS가 최신 버전

Java Virtual Machine

- ▶ Java는 직접 기계어로 컴파일하지 않고, 중간단계의 바이트 코드로 컴파일
- ▶ 바이트 코드를 읽고 수행할 수 있는 가상의 운영체제가 필요
 - ▶ JVM이 자바 프로그램과 실 운영체제를 중계
 - ▶ 한번의 작성으로 여러 운영체제와 상관 없는 프로그램 작성이 가능
- ▶ 이런 이유로 Java는 Compiler 언어와 Interpreter 언어의 혼합된 형태
 - ▶ 컴파일 단계에서 완전한 기계어로 번역되는 타 언어보다는 다소 느리다는 단점
 - ▶ JVM내 JIT(Just-In-Time) 컴파일러의 도입으로 속도의 격차는 많이 줄어듦

Java Programming

: 동작 방식



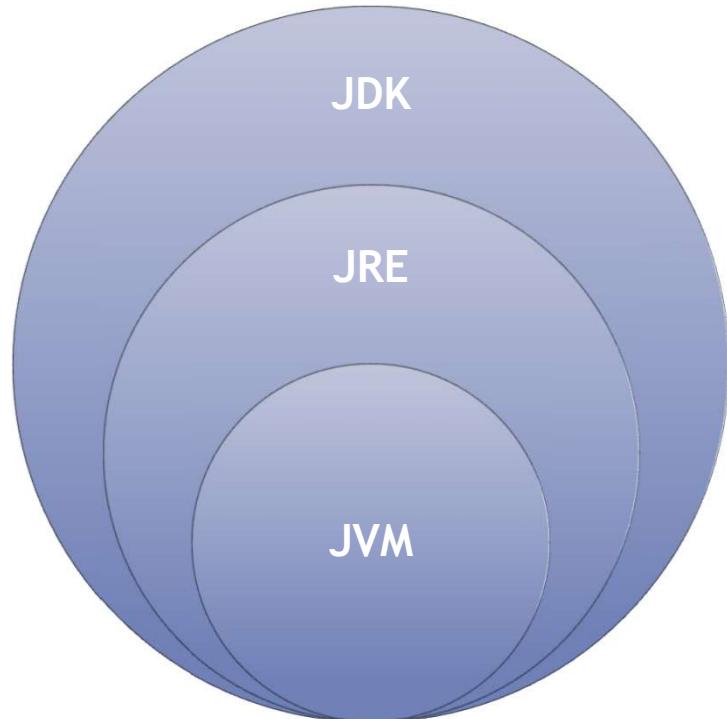
Java Programming

Development Environment

개발환경 구축

Java Development Environment

: JVM, JRE and JDK

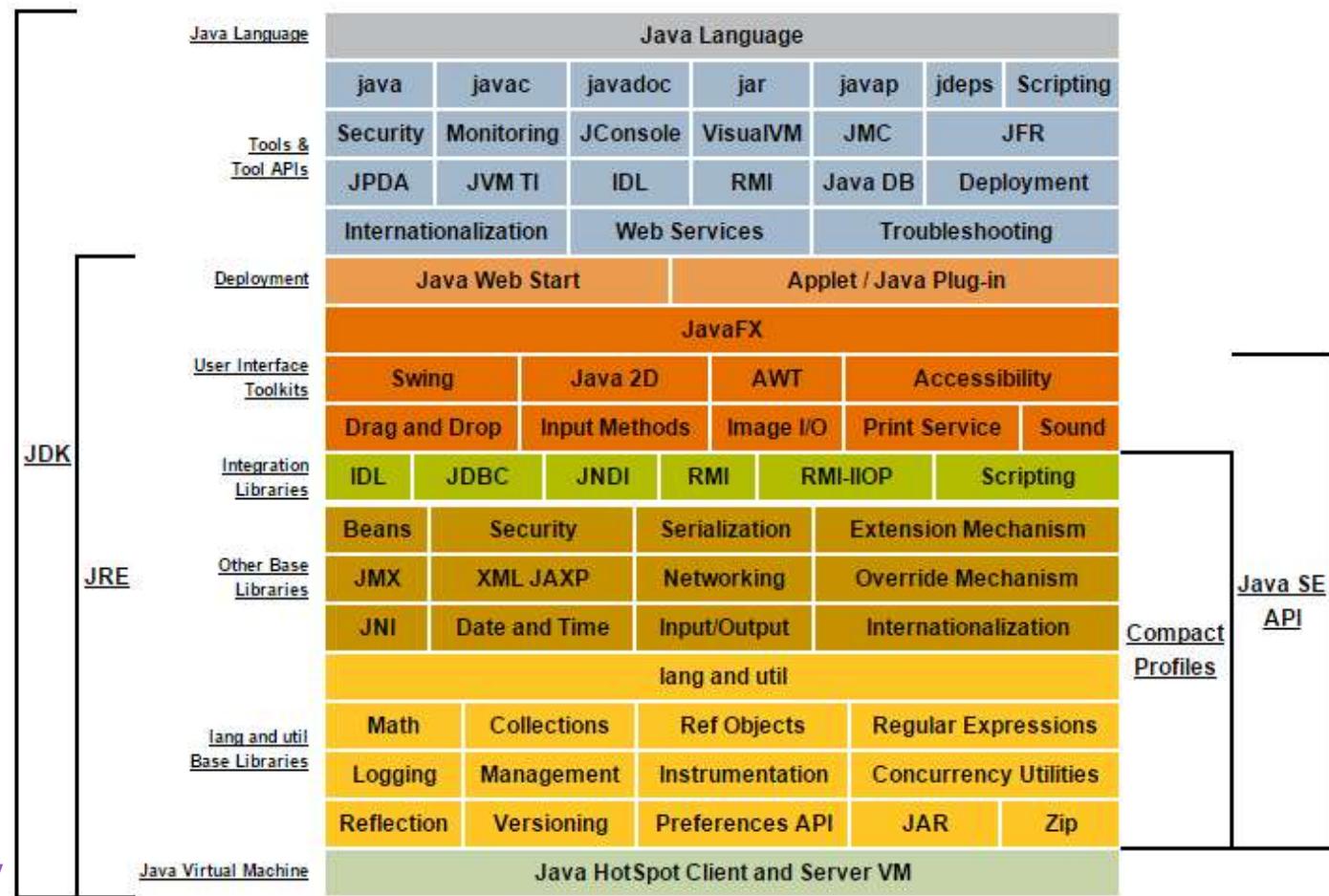


- ▶ JVM (Java Virtual Machine)
 - ▶ 자바 가상 머신
- ▶ JRE (Java Runtime Environment)
 - ▶ 자바 실행 환경
 - ▶ JVM + 표준 라이브러리
- ▶ JDK (Java Development Kit)
 - ▶ 자바 개발 키트
 - ▶ JRE + 라이브러리 API + 컴파일러 등
의 개발 도구

Java Development Environment

: Java Conceptual Diagram

Description of Java Conceptual Diagram



Java Development Environment

: Editions

JAVA의 분류

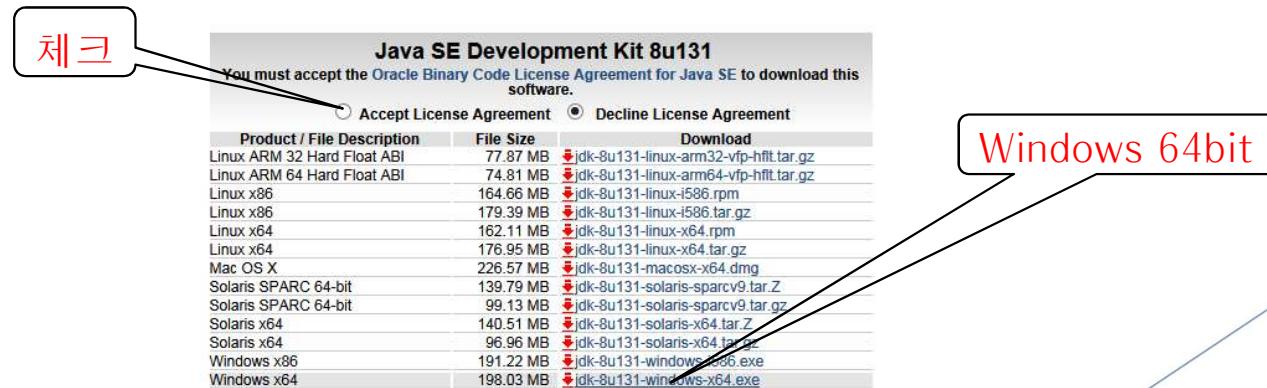
- ▶ SE (Standard Edition)
 - ▶ 일반 응용프로그램 개발용
- ▶ EE (Enterprise Edition)
 - ▶ 엔터프라이즈 응용프로그램 개발용
- ▶ ME (Micro Edition)
 - ▶ 소형 디바이스 개발용



Java Development Environment

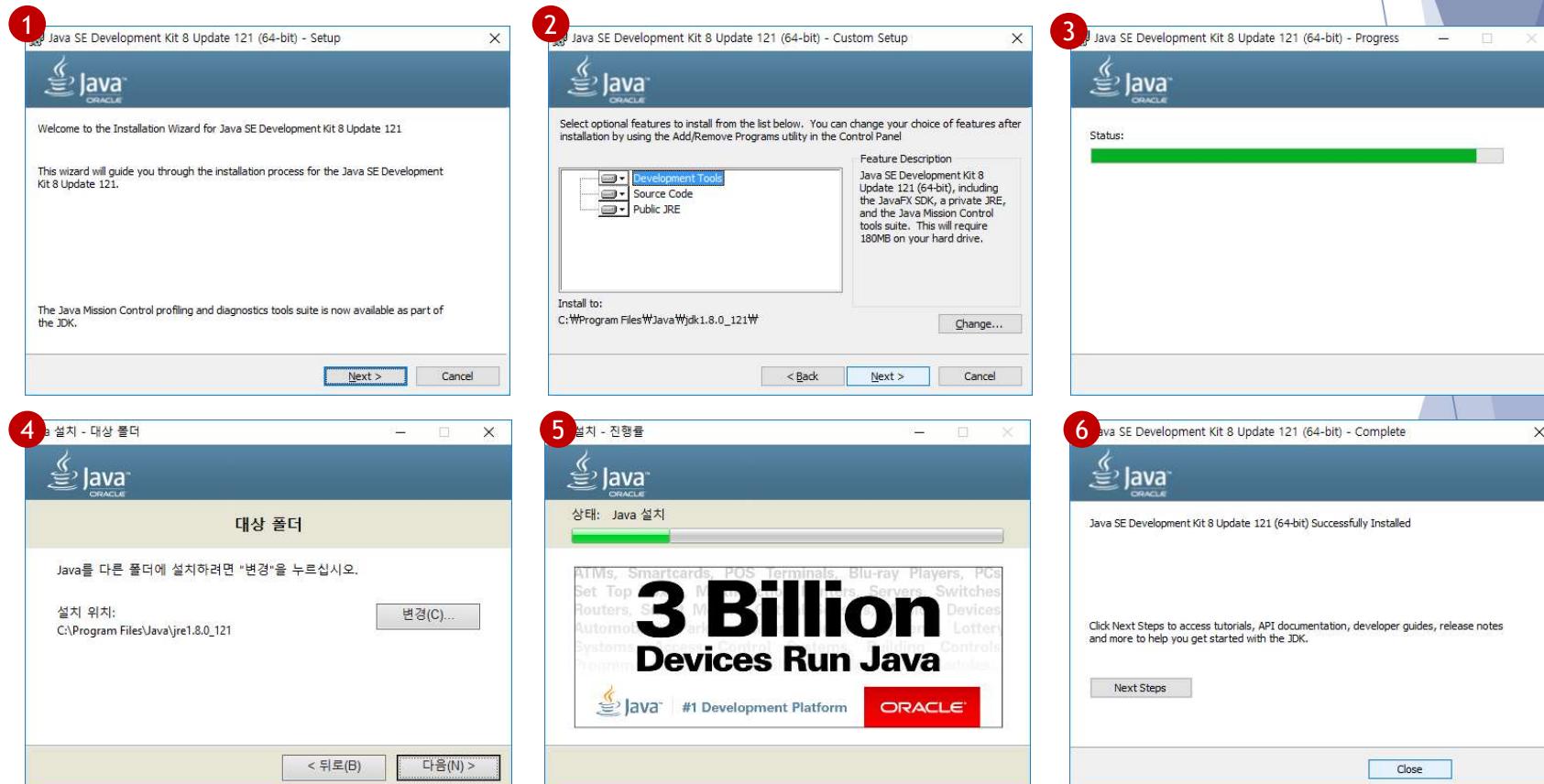
: Download JDK

- ▶ <http://java.oracle.com/>에서 다운로드
 - ▶ Downloads > Java SE > Downloads (탭)
 - ▶ License Agreement에 동의하고, 운영체제에 맞는 설치 파일을 다운로드



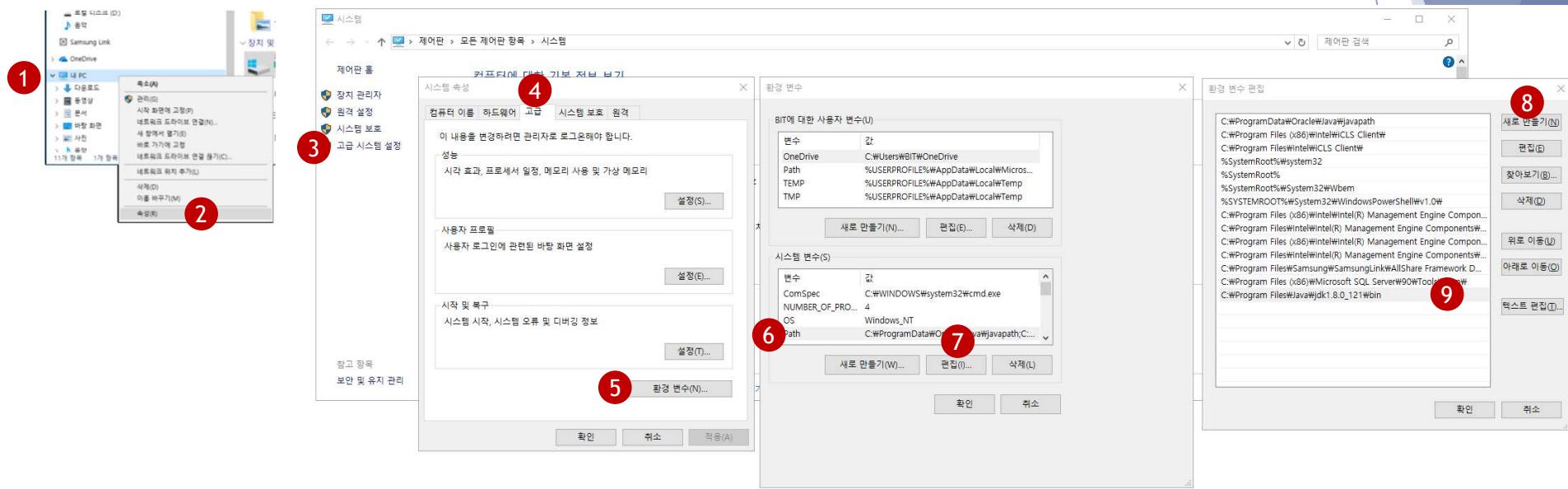
Java Development Environment

: Install JDK



Java Development Environment

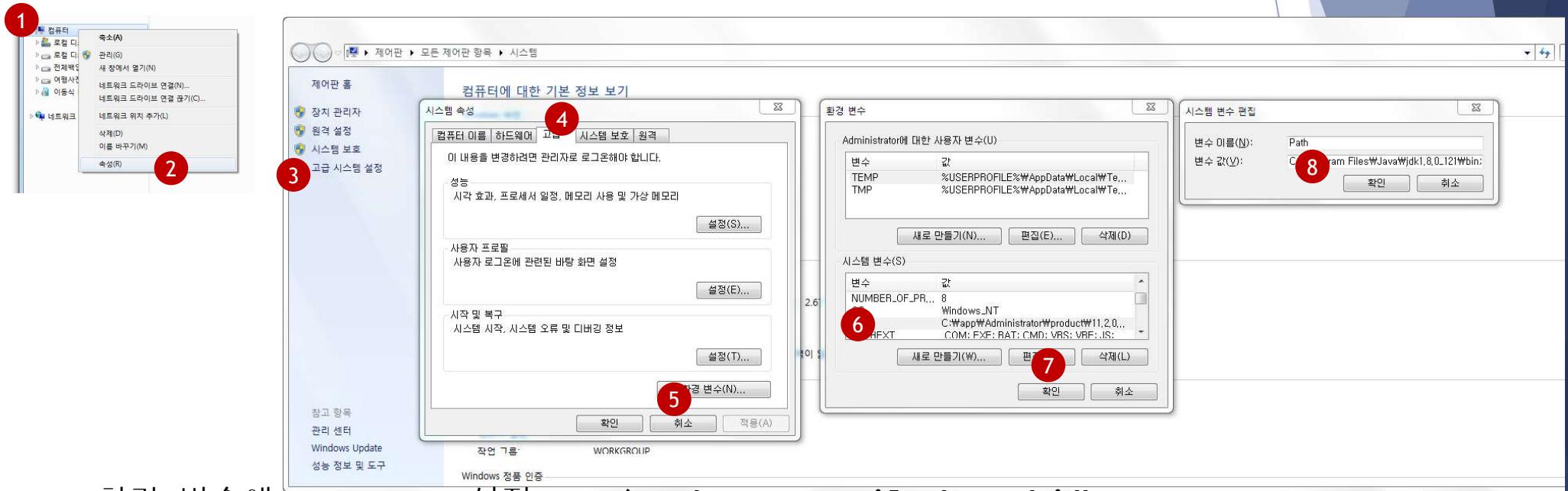
: 환경변수 설정 - Windows 8, 10



- 환경 변수에 JAVA_HOME 설정 : ex) C:\Program Files\Java\jdk1.8.0_131
- Path에 %JAVA_HOME%\bin 설정

Java Development Environment

: 환경변수 설정 - Windows 7



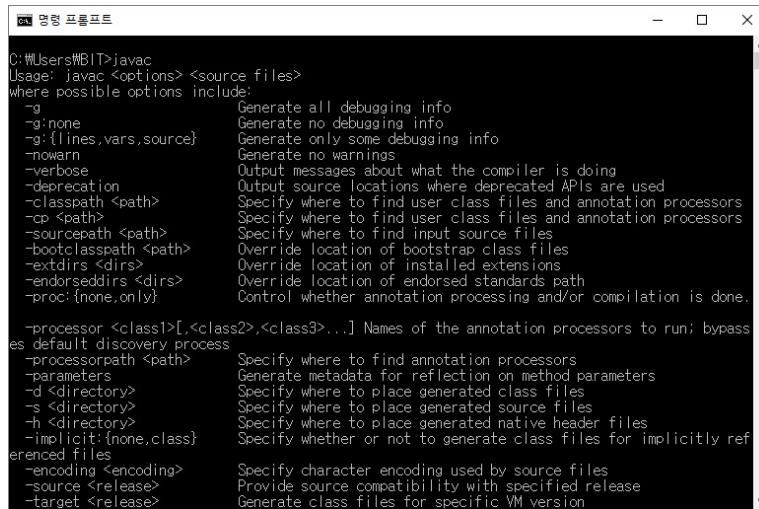
1. 환경 변수에 JAVA_HOME 설정 : ex) C:\Program Files\Java\jdk1.8.0_131
2. Path에 %JAVA_HOME%\bin 추가 :
ex) %JAVA_HOME%\bin;C:\Program Files (x86)\ATI Technologies\ATI.ACE\Core-Static

Java Development Environment

: 환경변수 설정 확인

▶ Win + R > cmd

javac 입력

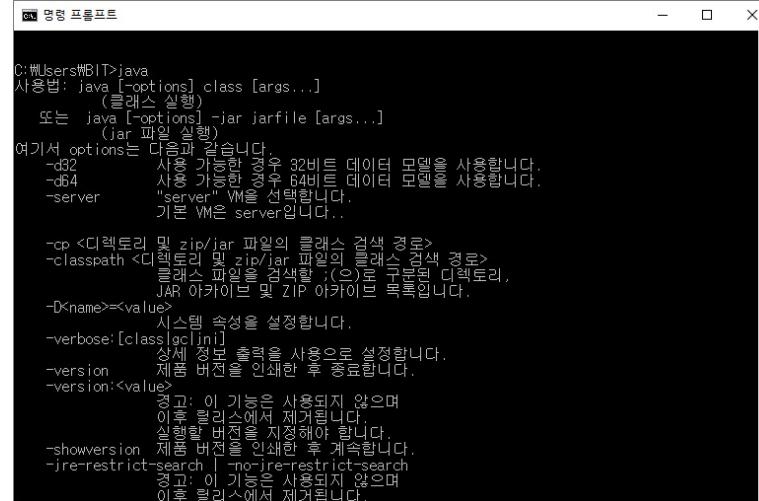


```
C:\Windows\system32>javac
Usage: javac [<options>] <source files>
where possible options include:
  -g           Generate all debugging info
  -g:none      Generate no debugging info
  -g:{lines,vars,source} Generate only some debugging info
  -nowarn      Generate no warnings
  -verbose     Output messages about what the compiler is doing
  -deprecation Output source locations where deprecated APIs are used
  -classpath <path> Specify where to find user class files and annotation processors
  -cp <path>   Specify where to find user class files and annotation processors
  -sourcepath <path> Specify where to find input source files
  -bootclasspath <path> Override location of bootstrap class files
  -extdirs <dirs> Override location of installed extensions
  -endorseddirs <dirs> Override location of endorsed standards path
  -proc:{none,only} Control whether annotation processing and/or compilation is done.

  -processor <class1>[,<class2>,<class3>...] Names of the annotation processors to run; bypass
es default discovery process
  -processorpath <path>  Specify where to find annotation processors
  -parameters        Generate metadata for reflection on method parameters
  -d <directory>    Specify where to place generated class files
  -s <directory>    Specify where to place generated source files
  -h <directory>    Specify where to place generated native header files
  -implied:{none,class}
  -encoding <encoding> Specify character encoding used by source files
  -source <release>  Provide source compatibility with specified release
  -target <release>  Generate class files for specific VM version
```

정상적으로 실행되지 않으면 PATH 설정을 재확인

java 입력



```
C:\Windows\system32>java
사용법: java [<options>] class [<args...>]
          (클래스 실행)
  또는  java [<options>] -jar jarfile [<args...>]
          (jar 파일 실행)
여기서 options는 다음과 같습니다.
  -d32      사용 가능한 경우 32비트 데이터 모델을 사용합니다.
  -d64      사용 가능한 경우 64비트 데이터 모델을 사용합니다.
  -server   "server" VM을 선택합니다.
            기본 VM은 server입니다.

  -cp <디렉토리 및 zip/jar 파일의 클래스 검색 경로>
  -classpath <디렉토리 및 zip/jar 파일의 클래스 검색 경로>
            클래스 파일을 검색할;(으)로 구분된 디렉토리,
            JAR 아카이브 및 ZIP 아카이브 복록입니다.

  -D<name>=<value>
            시스템 속성을 설정합니다.
  -verbose:[class|gc|jni]
            상세 정보 출력을 사용으로 설정합니다.
  -version  제품 버전을 인쇄한 후 종료합니다.
  -version:<value>
            경고: 이 기능은 사용되지 않으며
            이후 브라우저에서 제거됩니다.
            실행할 버전을 지정해야 합니다.
  -showversion 제품 버전을 인쇄한 후 계속합니다.
  -jre-restrict-search | -no-jre-restrict-search
            경고: 이 기능은 사용되지 않으며
            이후 브라우저에서 제거됩니다.
```

Java Development Environment

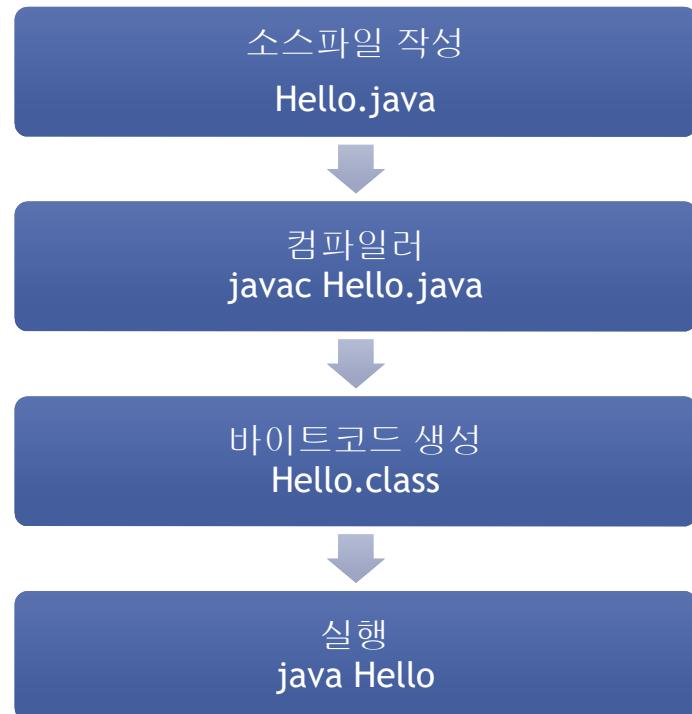
: JDK Document

The screenshot shows a web browser window displaying the Java Platform, Standard Edition 8 API Specification. The URL in the address bar is <http://docs.oracle.com/javase/8/docs/api/index.html>. The page title is "Java™ Platform, Standard Edition 8 API Specification". The left sidebar contains navigation links for "All Classes", "All Profiles", and "Packages". Under "Packages", it lists "java.applet", "java.awt", and "java.awt.color". The main content area includes sections for "Profiles" (with items "compact1", "compact2", "compact3") and "Packages" (with a table showing "java.applet" (Description: Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context), "java.awt" (Description: Contains all of the classes for creating user interfaces and for painting graphics and images), and "java.awt.color" (Description: Provides classes for color spaces)).

▶ <http://docs.oracle.com/javase/8/docs/api/index.html>

Java Development Environment

: 자바 개발 순서



Java Development Environment

: 첫 번째 자바 프로그램

- ▶ 소스 코드 작성
 - ▶ 편집기를 이용, 우측의 코드를 작성하고 “HelloWorld.java”라는 이름으로 저장
- ▶ 컴파일
 - ▶ 명령 프롬프트 실행
 - ▶ Hello.java 파일이 저장된 디렉토리로 이동하여 컴파일

```
javac HelloWorld.java
```

- ▶ 바이트코드 생성 확인
 - ▶ 컴파일한 디렉토리 내에서 class 파일 확인
- ▶ 실행하여 결과 확인

```
java HelloWorld
```

 }
}

Java Development Environment

: 주석문, 실행문과 세미콜론(;)

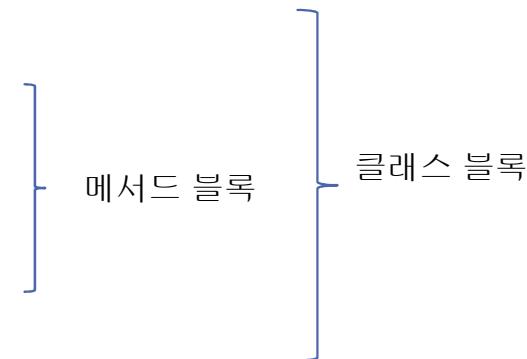
기호	설명
//	// 표시 지점부터 행 끝까지 주석으로 처리 (행 주석)
/* ~ */	/* 와 */ 사이 범위에 있는 내용을 주석으로 처리 (범위 주석)
/** ~ */	API 도큐먼트를 생성하는데 사용 (도큐먼트 주석)

- ▶ 실행문은 변수 선언, 값 저장, 메서드 호출에 해당하는 코드
- ▶ 실행문 맨 마지막에는 반드시 세미콜론(;)을 붙여야 함
 - ▶ 실행문이 끝났음을 자바에게 알려주는 표시

Java Development Environment

: 첫 번째 자바 프로그램 - 들여다보기

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



- ▶ 클래스 : 필드 또는 메서드를 포함하는 블록
- ▶ 메서드 : 어떤 일을 처리하는 실행문들을 모아 놓은 블록
- ▶ **public static void main(String[] args)** -> 익숙해질 때까지 연습

Java Development Environment

: Integrated Development Environment (IDE)



- ▶ 프로그램 개발에 관련된 모든 작업을 하나의 프로그램 안에서 처리하는 환경을 제공하는 소프트웨어
 - ▶ 코딩, 디버그, 컴파일, 배포 등
- ▶ 대표적인 Java IDE
 - ▶ Eclipse
 - ▶ Netbeans
 - ▶ IntelliJ IDEA

Java Development Environment

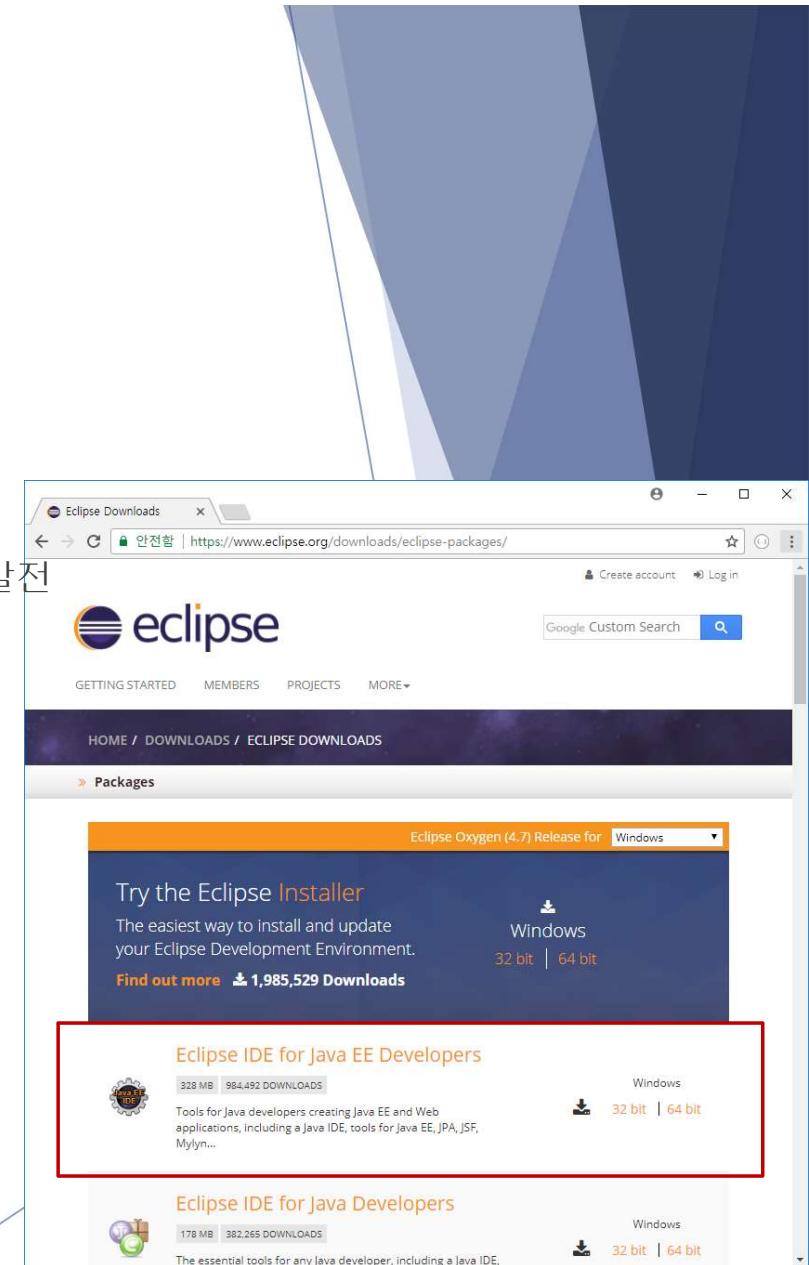
: Eclipse

▶ 특징

- ▶ 강력한 기능과 깔끔한 인터페이스
- ▶ IBM, 래쇼널 소프트웨어, 레드햇 등 여러 업체의 컨소시엄으로 개발, 발전
- ▶ 개발 목적에 맞는 다양한 버전과 변형이 있음
- ▶ 개발 편의를 위한 다양한 플러그인 제공
- ▶ 2017년 현재, 4.7 (Oxygen)이 최신 버전)

▶ 설치

- ▶ <https://www.eclipse.org/>에서 다운로드
- ▶ 개발 PC의 OS에 맞는 버전을 다운로드
- ▶ 다운로드 받은 파일을 원하는 위치에 압축 해제

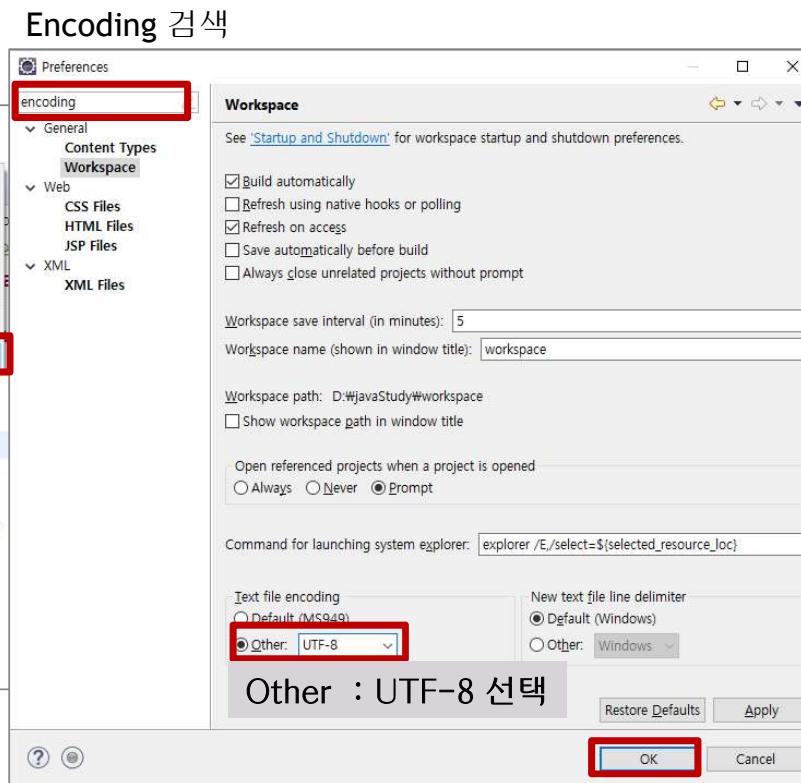
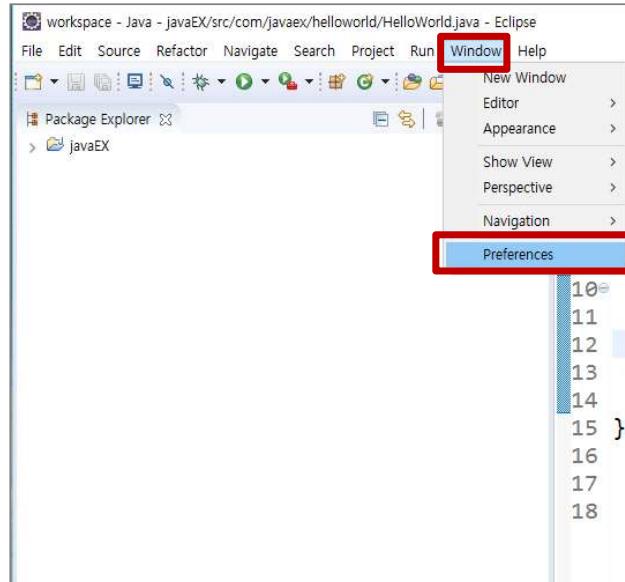


Java Development Environment

: Eclipse 설정

▶ Encoding 변경

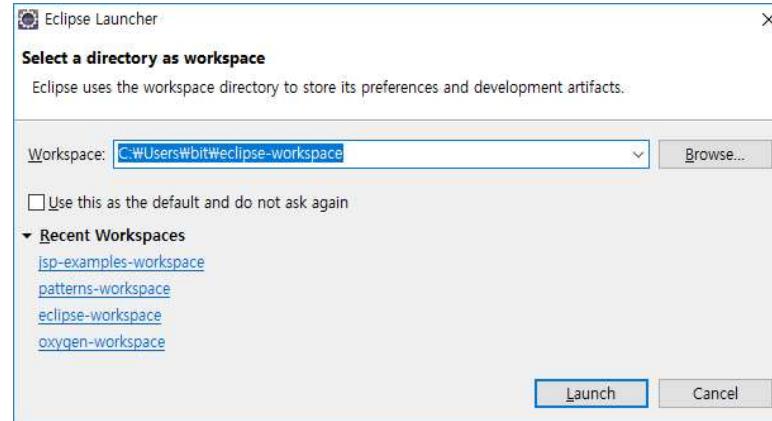
Windows > Preferences



Java Development Environment

: Eclipse 살펴보기

- ▶ 워크스페이스 (Workspace)
 - ▶ 이클립스에서 작성한 프로젝트가 저장되는 디렉토리
 - ▶ 이클립스를 사용하면서 변경되는 속성값들은 하위에 .metadata 디렉토리에서 관리
 - ▶ 원하면 다른 워크스페이스로 전환 할 수 있다.

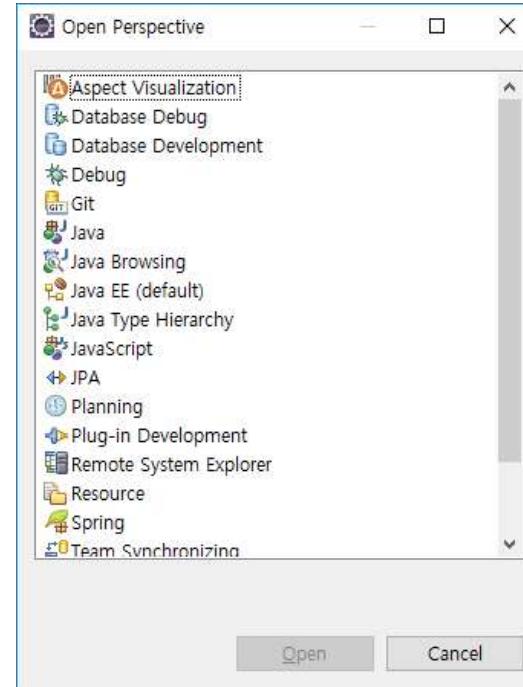


Java Development Environment

: Eclipse 살펴보기

▶ 퍼스펙티브 (Perspective)

- ▶ 프로젝트 개발시 유용하게 사용하는 뷰 (View)들을 묶어 놓은 것
- ▶ 개발하고자 하는 응용프로그램에 따라 적합한 뷰 배치를 해 둔 것

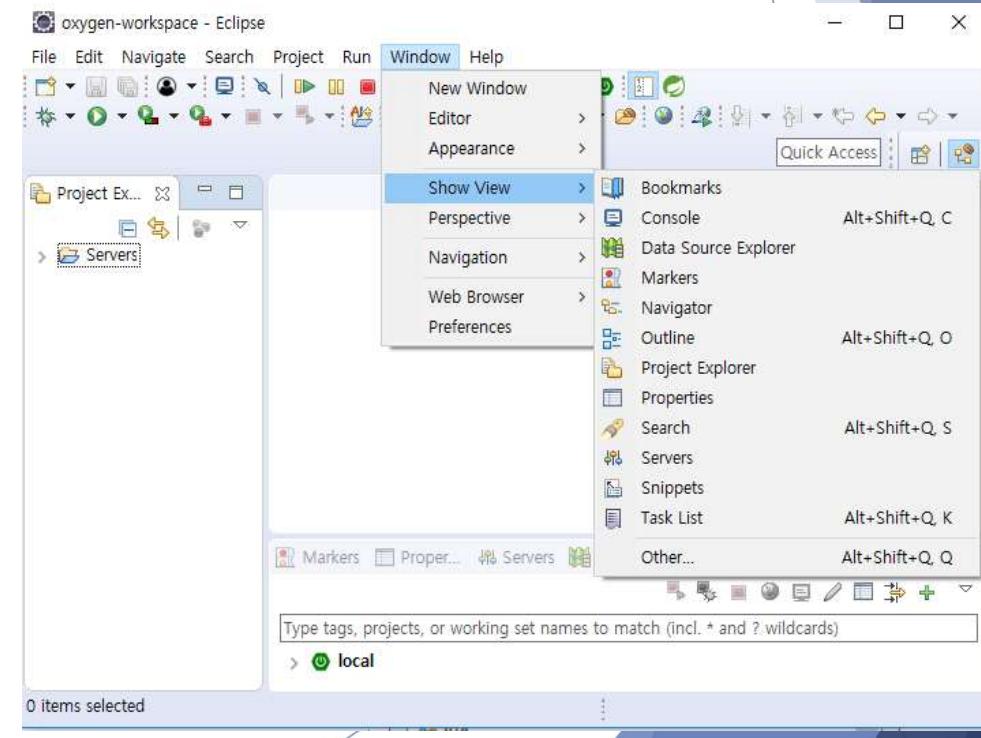


Java Development Environment

: Eclipse 살펴보기

▶ 뷰 (View)

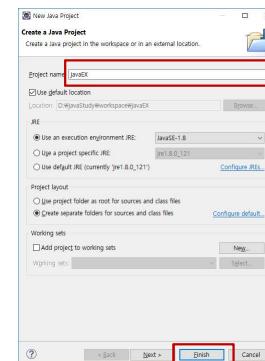
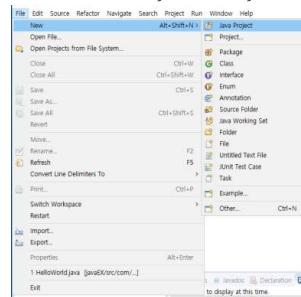
- ▶ 이클립스 내부에서 사용되는 용도별 창
- ▶ 원하는 뷰가 보이지 않는다면, Window > Show View 메뉴에서 찾는다



Java Development Environment

: 첫 번째 Eclipse 프로젝트

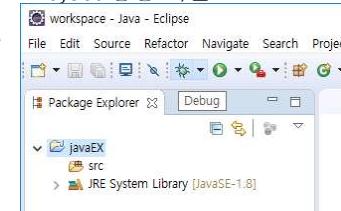
File > New > java Project 선택



javaEx 입력

finish 버튼 클릭

Project 생성 확인



프로젝트 만들기

패키지 만들기

클래스 만들기

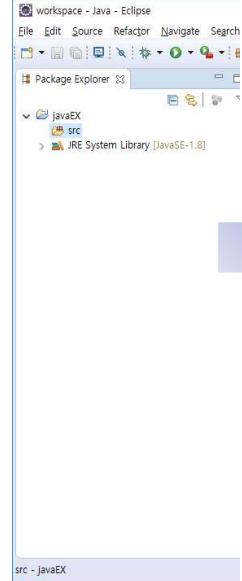
소스코드 작성

실행

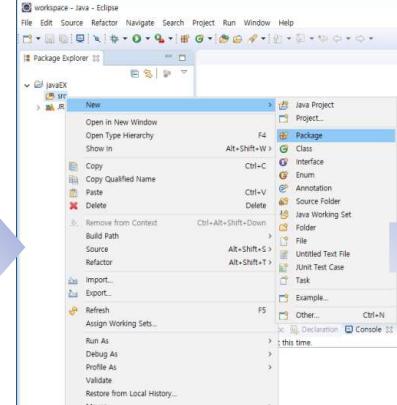
Java Development Environment

: 첫 번째 Eclipse 프로젝트

src 폴더 선택 후
오른쪽 마우스 클릭

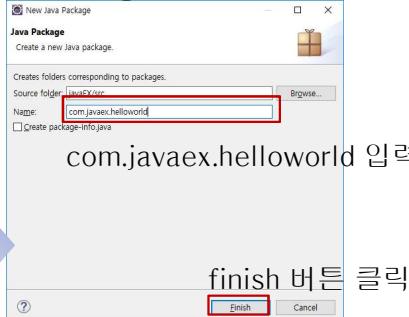


New > Package 선택



Package명 입력

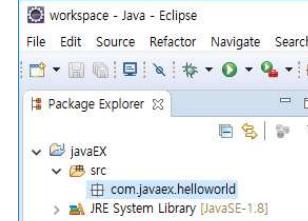
*Package명은 소문자 사용



com.javaex.helloworld 입력

finish 버튼 클릭

Package 생성 확인



프로젝트 만들기

패키지 만들기

클래스 만들기

소스코드 작성

실행

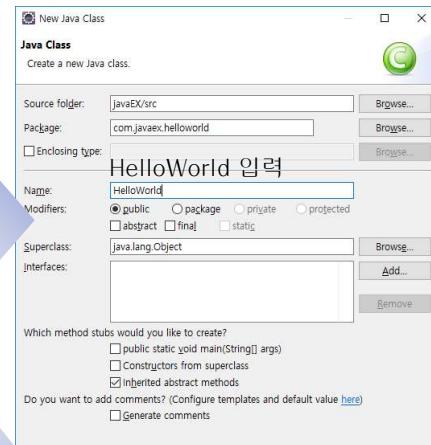
Java Development Environment

: 첫 번째 Eclipse 프로젝트

com.javaex.helloworld
패키지 선택 후
오른쪽 마우스 클릭

New > Class 선택

Class명 입력
*Class 명은 첫글자를 대문자로 사용



finish 버튼 클릭

프로젝트 만들기

패키지 만들기

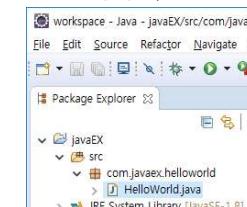
클래스 만들기

소스코드 작성

실행

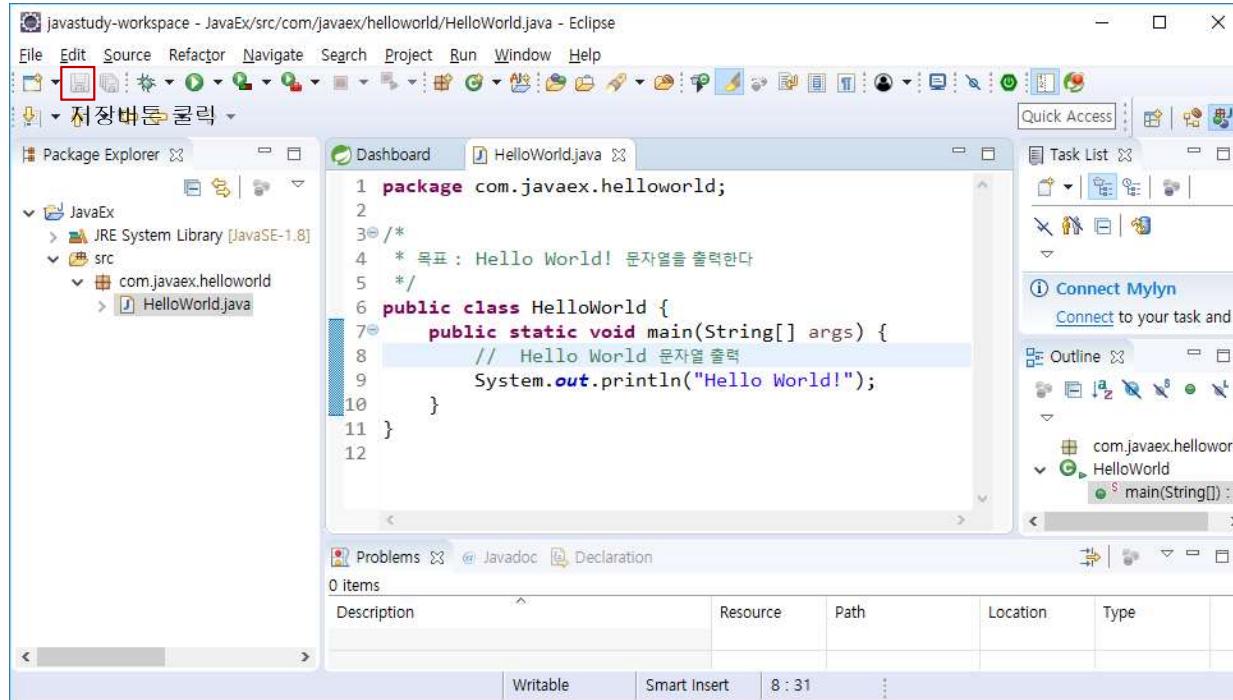


Class 생성 확인



Java Development Environment

: 첫 번째 Eclipse 프로젝트



프로젝트 만들기

패키지 만들기

클래스 만들기

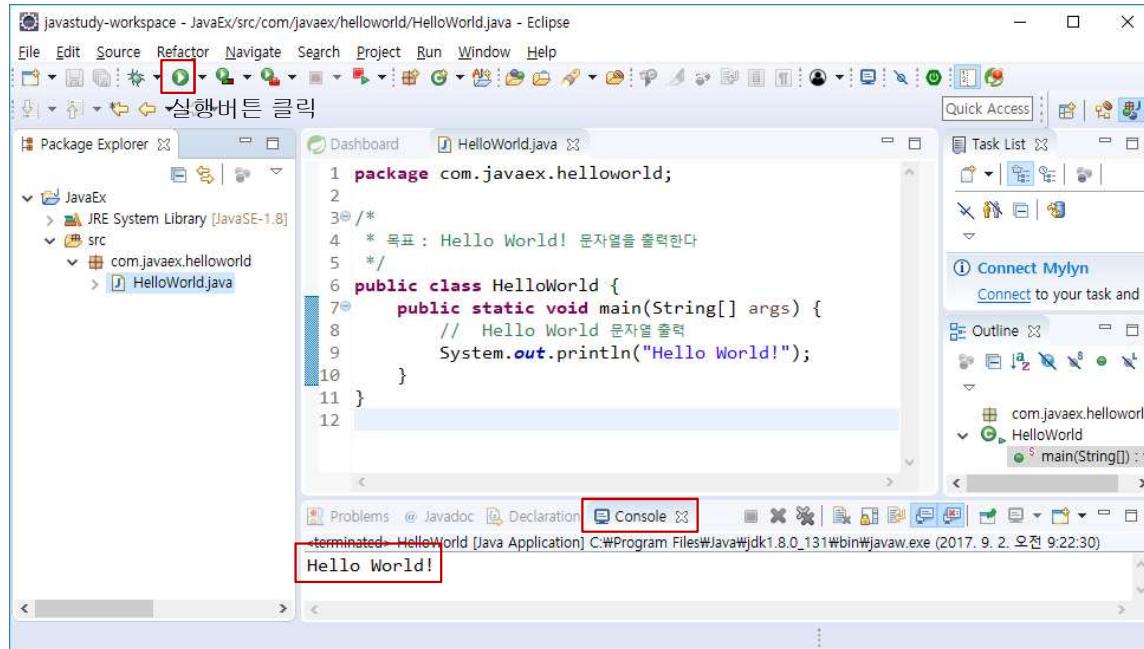
소스코드 작성

실행

Java Development Environment

: 첫 번째 Eclipse 프로젝트

실행버튼 클릭 후 console창 결과 확인



프로젝트 만들기

패키지 만들기

클래스 만들기

소스코드 작성

실행

Java Development Environment

: 코드 구조 살펴보기

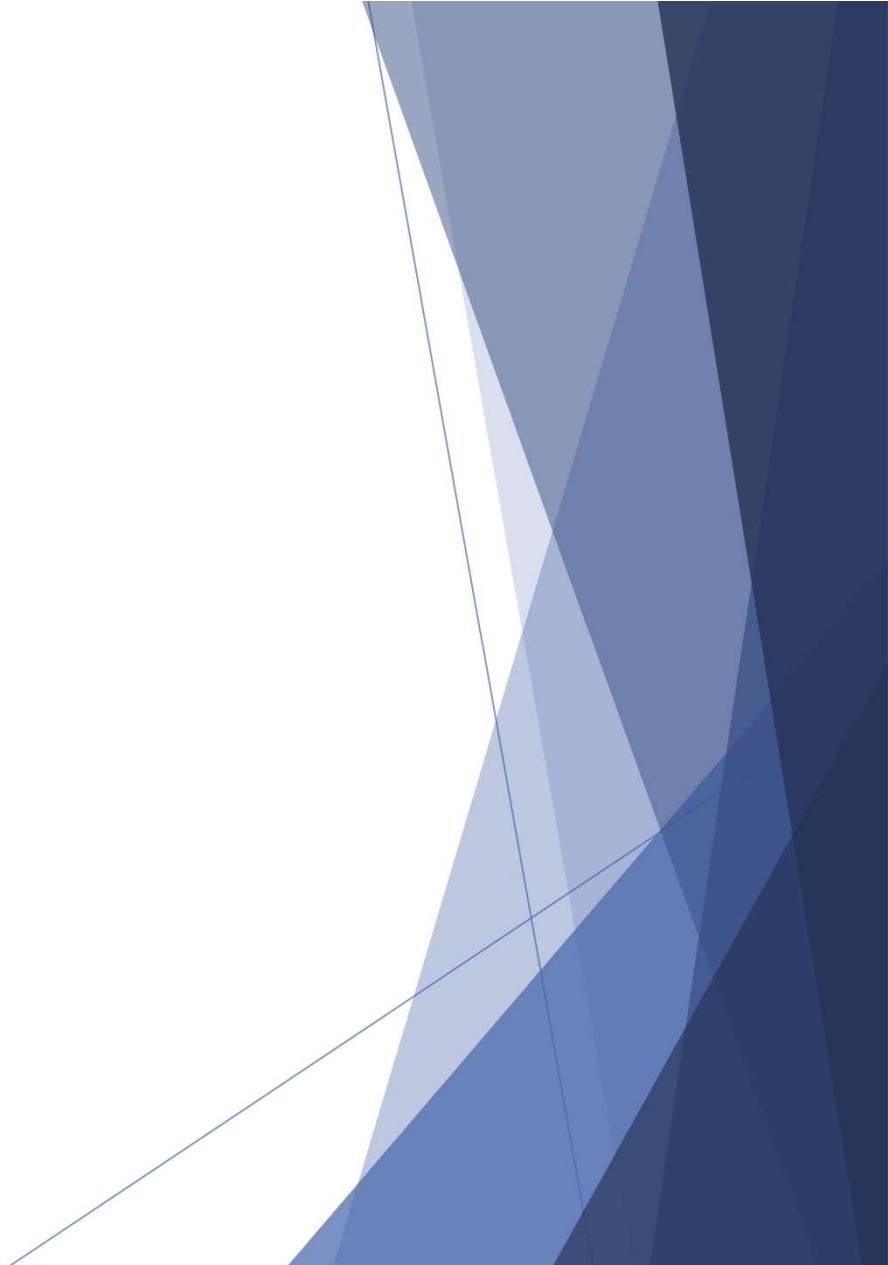
```
① package com.javaex.helloworld;  
  
/* ②  
 * 목표 : Hello World! 문자열을 출력한다  
 */  
  
public class HelloWorld {  
    public static void main(String[] args) {  
        // Hello World 문자열 출력 ⑤  
        ③ ④ ⑥ System.out.println("Hello World!"); ⑦  
    }  
}
```

- (1) 패키지 경로, 패키지명
클래스를 체계적으로 관리하기 위한 묶음
소문자로 작성
- (2) 범위 주석
코드를 설명하기 위한 문장
프로그램 수행에는 아무 영향 없음
- (3) 클래스 블록
클래스명의 첫글자는 대문자로 작성
파일명과 클래스명은 일치
- (4) 메서드 블록
여러 실행문의 묶음
내부의 실행문을 절차적으로 실행
- (5) 행 주석
- (6) 실행문
- (7) 문장의 끝은 세미콜론(;)으로 표시

Java Programming

Language
언어

예약어와 식별자



Java Language

: 예약어와 식별자

▶ 예약어 (Keyword)

- ▶ 프로그래밍 언어에 미리 정의된 단어
- ▶ 식별자로 사용하지 않음

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

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

Java Language

: 예약어와 식별자

▶ 식별자 (Identifier)

- ▶ 프로그래머가 직접 만들어주는 이름
- ▶ 변수명, 클래스명, 메서드명 등
- ▶ 명명 규칙(Naming Convention)에 따라 지정

작성 규칙

문자, \$, _로 시작해야 함

숫자로 시작할 수 없음

대소문자가 구분됨

예약어는 사용할 수 없음

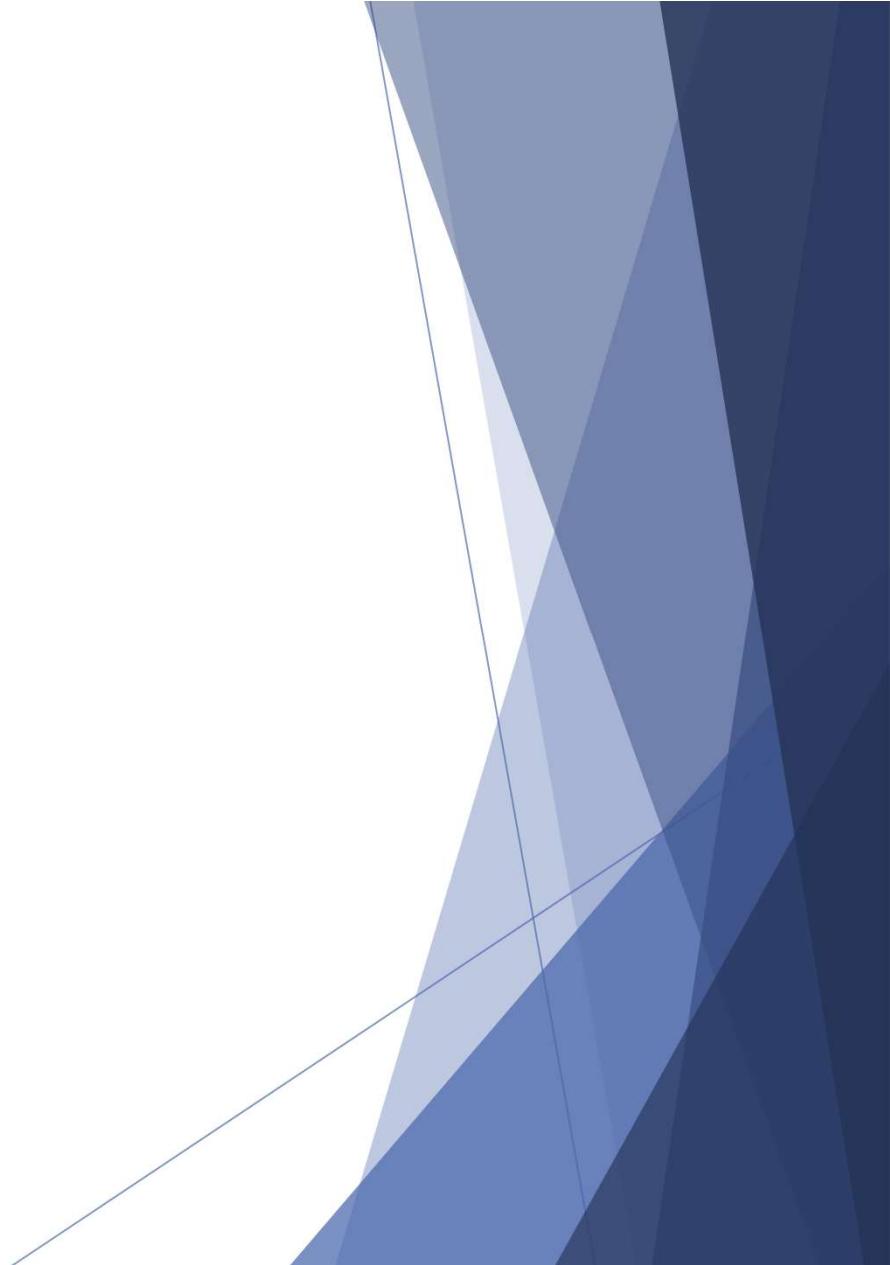
[식별자 사용 예시]

```
/*
작성자:남승균
작성일:2017.04.03
설 명:Hello World 출력하기
*/
package com.javaex.helloworld;

public class HelloWorld {

    public static void main(String[] args) {
        //Hello World 를 출력합니다.
        System.out.println("Hello World!");
    }
}
```

변수와 자료형



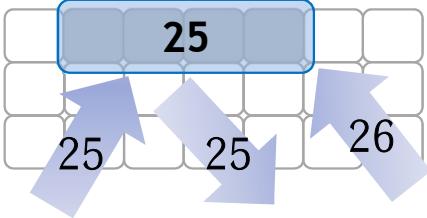
Java Language

: 변수와 자료형

▶ 변수 (Variable)

- ▶ 값(데이터)을 저장하기 위한 메모리 공간
- ▶ 하나의 변수는 하나의 자료형만 지정할 수 있음
- ▶ 값을 저장하고 조회하고 변경할 수 있음

[변수사용예시]

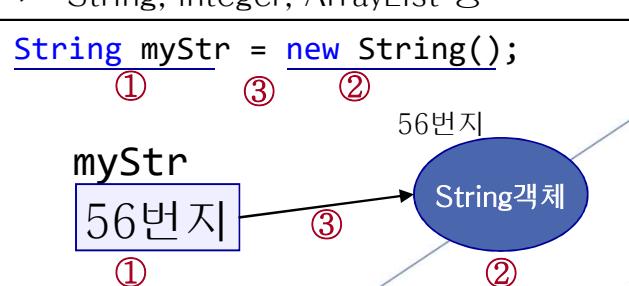
-개념-	-메모리-	-코드-
내 나이를 저장할래 내 나이는 25 내 나이는? 나이를 26 으로 변경	<p>myAge</p>  <p>25</p> <p>25</p> <p>26</p>	<pre>int myAge; // 선언 myAge = 25; // 초기화 System.out.println(myAge); // 조회 myAge = 26; // 변경</pre>

Java Language

: 변수와 자료형

▶ 자료형 (Type)

- ▶ 자료형에 따라 저장할 수 있는 값을 종류와 범위가 결정
- ▶ 변수를 사용하는 도중 변경할 수 없음

기본자료형 (primitive data types)	참조자료형 (reference data types)
<ul style="list-style-type: none">✓ 최소 단위의 자료형 다른 자료형으로 분해되지 않음✓ 메소드 없이 값만 가짐✓ int, float, double, char 등	<ul style="list-style-type: none">✓ 여러 자료형들의 집합 구성된 클래스의 객체를 참조✓ 데이터와 메소드를 가짐✓ String, Integer, ArrayList 등
<pre>int myInt = 19;</pre> <p>myInt 19</p>	<pre>String myStr = new String();</pre> <p>① ③ ②</p> <p>myStr 56번지</p> <p>56번지 → String객체 ① ③ ②</p> 

Java Language

: 변수와 자료형

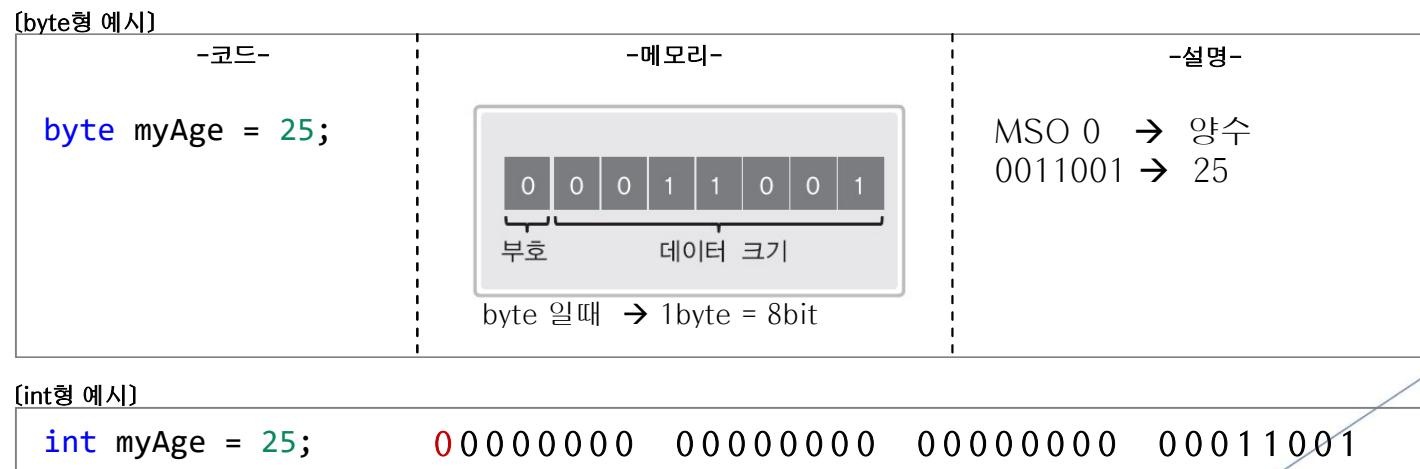
▶ 기본 자료형(primitive data types)

자료형	키워드	크기	표현 범위	사용 예
논리형	boolean	1byte	true OR false(0과 1이 아니다)	<code>boolean isFun = true;</code>
문자형	char	2byte	모든 유니코드 문자 (\u0000~\uFFFF, 0~65535)	<code>char c = 'f';</code>
정수형	byte	1byte	-128 ~ 127	<code>byte b = 89;</code>
	short	2byte	-32,768 ~ 32,767	<code>short s = 32760;</code>
	int	4byte	-2,147,483,648 ~ 2,147,483,647	<code>int x = 59;</code>
	long	8byte	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807	<code>long big = 345678912345L;</code>
실수형	float	4byte	-3.4E38 ~ 3.4E38	<code>float f = 32.5F;</code>
	double	8byte	-1.7E308 ~ 1.7E308	<code>double d = 2.3e10;</code>

Java Language

: 기본 자료형 - short, int, long (정수형)

- ▶ 가장 왼쪽 비트(MSB: Most Significant Bit)는 부호를 표시하는 비트
 - ▶ MSB를 제외한 나머지는 데이터의 크기를 나타냄
 - ▶ 처리할 수 있는 수의 범위에 따라 **short < int < long**으로 구분



Java Language

: 기본 자료형 - float, double (실수형)

- ▶ 정수형 자료형과 값 저장방식이 다름 : 더 큰 값을 저장할 수 있음
- ▶ 실수형 저장을 위한 기본식
- ▶ $(+/-) m * 10^n$

IEEE Floating Point Representation



IEEE Double Precision Floating Point Representation



Java Language

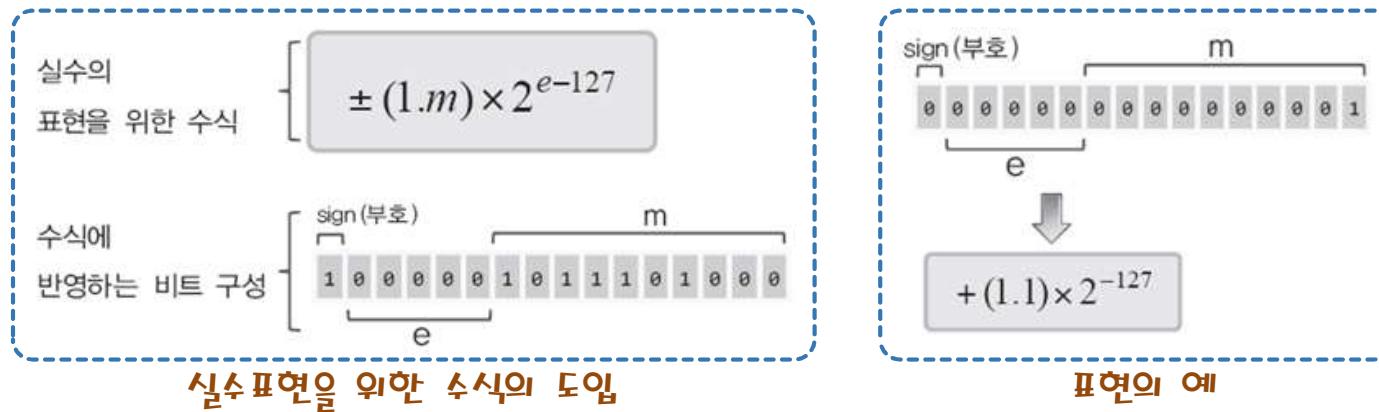
: 기본 자료형 - float, double (실수형)

- ▶ 실수 표현의 문제

- ▶ 0과 1사이의 실수만 해도 그 수가 무한대
- ▶ 바이트 추가로 모든 수의 표현은 불가능

- ▶ Solution:

- ▶ 정밀도를 포기하고 표현할 수 있는 값의 범위를 넓하자



Java Language

: 기본 자료형 - boolean (논리형)

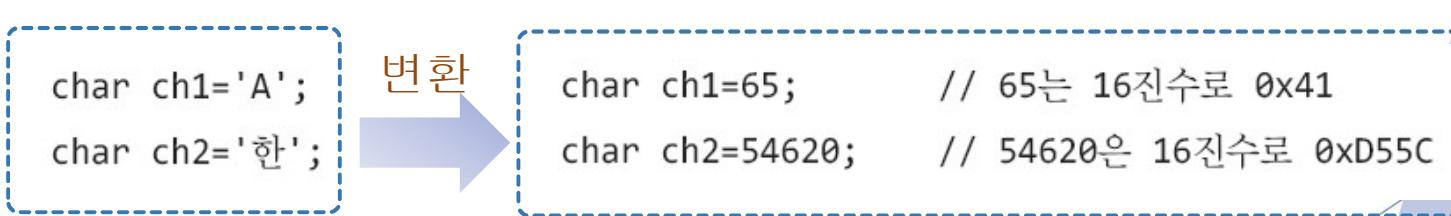
- ▶ 논리값(true/false)을 저장하는 자료형
- ▶ 값에 따라 조건문과 제어문의 흐름을 변경할 때 사용
 - ▶ true : ‘참’을 의미하는 키워드 혹은 조건을 만족함
 - ▶ false : ‘거짓’을 의미하는 키워드 혹은 조건을 만족하지 않음
- ▶ 키워드 true와 false의 이해
 - ▶ 숫자의 관점에서 이해하지 말자
 - ▶ Java에서 true/false는 그 자체로 저장이 가능한 데이터

Java Language

: 기본 자료형 - char (문자형)

▶ 문자형

- ▶ 문자 하나를 유니코드 기반으로 표현
- ▶ 유니코드 : 전 세계의 문자를 표현할 수 있는 국제표준 코드 집합
- ▶ 문자는 작은 따옴표(')로 묶어서 지정
- ▶ 문자는 **char**형 변수에 저장한다. 저장시 실제는 유니코드 값 저장



Java Language

: 기본 자료형 - char(문자형)

	00	01	02	03	04	05	06	0	AC0	AC1	AC2	AC3	AC4	AC5	AC6	AC7	AC8
00	<u>NUL</u> 0000	<u>STX</u> 0001	<u>SOT</u> 0002	<u>ETX</u> 0003	<u>EOT</u> 0004	<u>ENQ</u> 0005	<u>ACI</u> 0006	0	가 AC00	감 AC10	캔 AC20	겠 AC30	걀 AC40	각 AC50	값 AC60	거 AC70	검 AC80
10	<u>DLE</u> 0010	<u>DC1</u> 0011	<u>DC2</u> 0012	<u>DC3</u> 0013	<u>DC4</u> 0014	<u>NAK</u> 0015	<u>SYN</u> 0016	1	각 AC01	갑 AC11	갯 AC21	갱 AC31	갉 AC41	같 AC51	깁 AC61	걱 AC71	겁 AC81
20	<u>SP</u> 0020	!	"	#	\$	%	&		깎 AC02	값 AC12	깁 AC22	깟 AC32	깁 AC42	깁 AC52	깁 AC62	꺾 AC72	깁 AC82
30	0	1	2	3	4	5	6	2	깎 AC03	갓 AC13	갠 AC23	깟 AC33	깁 AC43	깁 AC53	깁 AC63	꺾 AC73	깁 AC83
40	©	A	B	C	D	E	F		값 AC04	갓 AC14	갠 AC24	깟 AC34	깁 AC44	깁 AC54	깁 AC64	꺾 AC74	깁 AC84
50	P	Q	R	S	T	U	V	3	값 AC05	갓 AC15	갠 AC25	깟 AC35	깁 AC45	깁 AC55	깁 AC65	꺾 AC75	깁 AC85
60	~	a	b	c	d	e	f		값 AC06	갓 AC16	갠 AC26	깟 AC36	깁 AC46	깁 AC56	깁 AC66	꺾 AC76	깁 AC86
70	p	q	r	s	t	u	v		값 AC07	갓 AC17	갠 AC27	깟 AC37	깁 AC47	깁 AC57	깁 AC67	꺾 AC77	깁 AC87

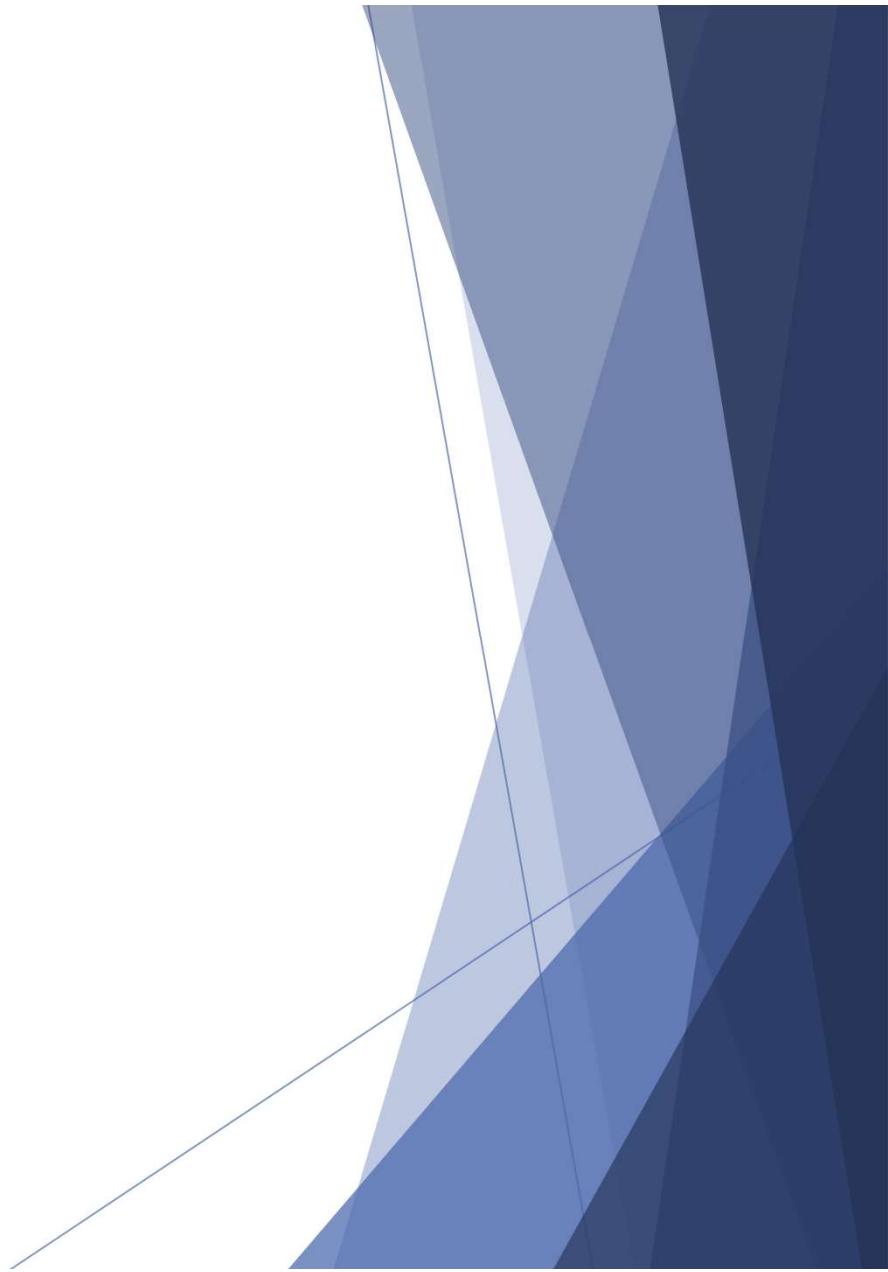
연습문제

: 변수와 자료형

- ▶ 괄호 안에 적절한 데이터 타입을 기술하시오

```
(    ) number;      // 학번  
(    ) name;        // 이름  
(    ) isEnrolled; // 등록 여부  
(    ) grade;       // 평점  
(    ) address;     // 주소  
(    ) major;        // 전공  
(    ) unit;         // 이수 학점  
(    ) haveMinor;   // 부전공 여부  
(    ) juminNo;     // 주민번호(-없이 13자리숫자)  
(    ) cellphone;   // 핸드폰 번호(-포함한 숫자)  
(    ) age;          // 나이  
(    ) email;        // 이메일주소
```

상수와 형변환



Java Language

: 상수 (Constant)

- ▶ 변경할 수 없는 고정된 데이터
 - ▶ 할당, 조회는 되나 변경은 할 수 없음
- ▶ 코드의 이해와 변경이 쉬움
- ▶ 정의
 - ▶ static final로 선언
 - ▶ 대문자로 표현 (관례)
 - ▶ 여러 단어가 겹칠 경우 _로 구분

상수 선언 예제

```
static final double PI = 3.14159;  
static final int MAXIMUM_SPEED = 110;
```

Java Language

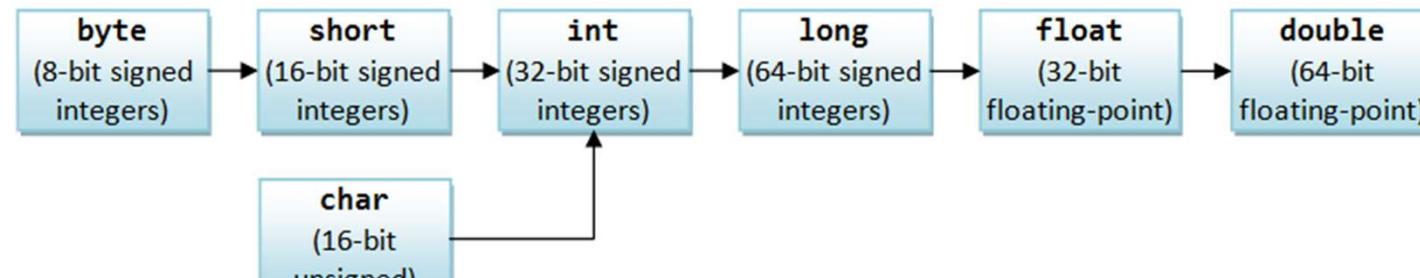
: 형 변환 (Type Casting)

▶ 암묵적 형 변환 (자동 타입 변환: Promotion)

- ▶ 자료의 범위가 좁은 자료형에서 넓은 자료형으로의 변환은 시스템이 자동으로 행함

암묵적 형변환

```
int num1 = 2;  
float num2 = 1.2F;  
float multiply = num1 * num2;
```



Orders of Implicit Type-Casting for Primitives

Java Language

: 형 변환 (Type Casting)

▶ 명시적 형 변환 (강제 타입 변환: Casting)

- ▶ 자료의 범위가 넓은 자료형에서 좁은 자료형으로 변환은 프로그래머가 강제로 변환해야 함
- ▶ 이때, 자료의 유실이 일어날 수 있으므로 주의해야 함

[정상변환X]

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

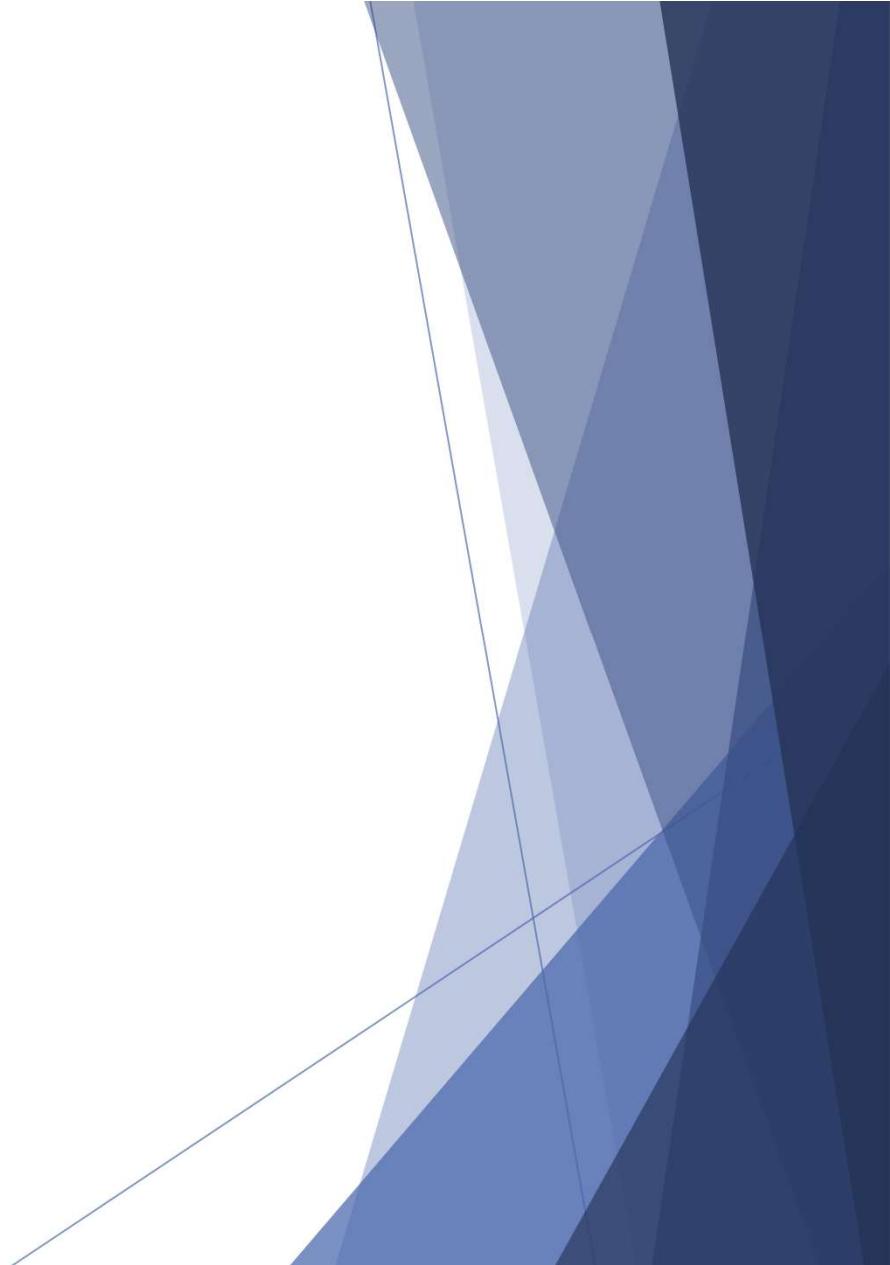
00000110 00100100 00011100 00001010 → 10

[정상변환]

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

00000000 00000000 00000000 00001010 → 10

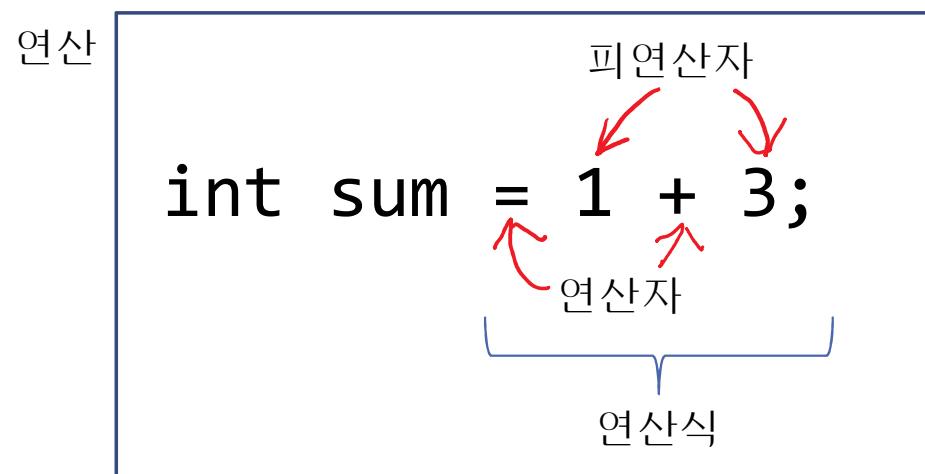
연산자



Java Language

: 연산자 (Operator)

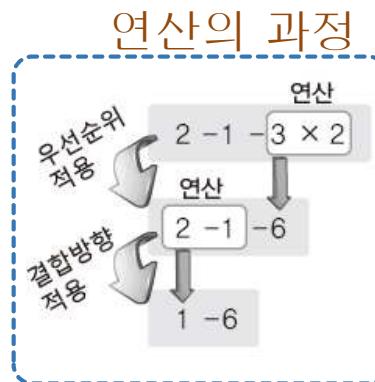
- ▶ 연산 (Operation) : 데이터를 처리하여 결과를 산출하는 것
- ▶ 연산자 (Operator) : 연산에 사용되는 표시나 기호 (데이터를 처리하는 기능을 수행)
- ▶ 피연산자 (Operand) : 연산 되는 데이터
- ▶ 연산식 (Expression) : 연산자와 피연산자를 이용, 연산 과정을 기술한 것



Java Language

: 연산자 우선순위

연산기호	결합방향	우선순위
[], .	→	1(높음)
expr++, expr--	←	2
++expr, --expr, +expr, -expr, ~, !, (type)	←	3
*, /, %	→	4
+, -	→	5
<<, >>, >>>	→	6
<, >, <=, >=, instanceof	→	7
==, !=	→	8
&	→	9
^	→	10
	→	11
&&	→	12
	→	13
? expr : expr	←	14
=, +=, -=, *=, /=, %=, &=, ^=, =, <<=, >>=	←	15(낮음)



Java Language

: 대입연산자(=)와 산술연산자(+, -, *, /, %)

연산자	연산자의 기능	결합방향
=	연산자 오른쪽에 있는 값을 연산자 왼쪽에 있는 변수에 대입한다. 예) val = 20;	◀
+	두 피연산자의 값을 더한다. 예) val = 4 + 3;	▶
-	왼쪽의 피연산자 값에서 오른쪽의 피연산자 값을 뺀다. 예) val = 4 - 3;	▶
*	두 피연산자의 값을 곱한다. 예) val = 4 * 3;	▶
/	왼쪽의 피연산자 값을 오른쪽의 피연산자 값으로 나눈다. 예) val = 7 / 3;	▶
%	왼쪽의 피연산자 값을 오른쪽의 피연산자 값으로 나눴을 때 얻게 되는 나머지를 반환한다. 예) val = 7 % 3	▶

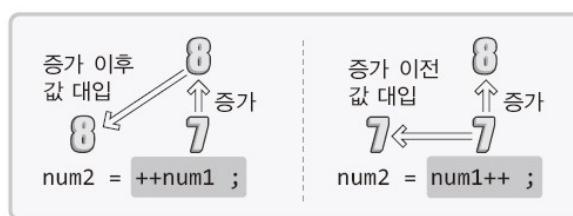
Java Language

: 증가, 감소 연산자

값의 변경 순서에 유의하자

연산자	연산자의 기능	결합방향
++ (prefix)	피연산자에 저장된 값을 1 증가 예) val = ++n;	◀
-- (prefix)	피연산자에 저장된 값을 1 감소 예) val = --n;	◀

연산자	연산자의 기능	결합방향
++ (postfix)	피연산자에 저장된 값을 1 증가 예) val = n++;	◀
-- (postfix)	피연산자에 저장된 값을 1 감소 예) val = n--;	◀



Java Language

: 관계 연산자

연산자	연산자의 기능	결합방향
<	예) $n1 < n2$ $n1이 n2보다 작은가?$	→
>	예) $n1 > n2$ $n1이 n2보다 큰가?$	→
\leq	예) $n1 \leq n2$ $n1이 n2보다 같거나 작은가?$	→
\geq	예) $n1 \geq n2$ $n1이 n2보다 같거나 큰가?$	→
$==$	예) $n1 == n2$ $n1과 n2가 같은가?$	→
$!=$	예) $n1 != n2$ $n1과 n2가 다른가?$	→

Java Language

: 논리 연산자

- ▶ AND (`&&`), OR (`||`), NOT (`!`)
- ▶ 결과는 boolean 타입

a	b	<code>a&& b</code>	<code>a b</code>	<code>!a</code>
False	False	False	False	True
False	True	False	True	
True	False	False	True	False
True	True	True	True	

Java Language

: 비트 연산자

- 정수형 데이터를 비트 단위로 개별 조작

비트 A	비트 B	비트 A & 비트 B
1	1	1
1	0	0
0	1	0
0	0	0

비트 A	비트 B	비트 A 비트 B
1	1	1
1	0	1
0	1	1
0	0	0

비트 A	비트 B	비트 A ^ 비트 B
1	1	0
1	0	1
0	1	1
0	0	0

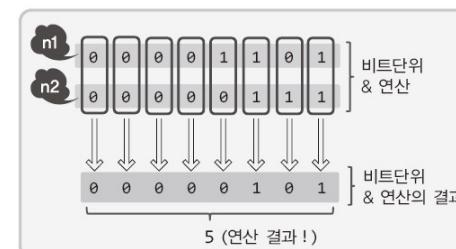
비트	~비트
1	0
0	1

~(NOT)

&(and)

|(OR)

반만true
^(XOR)



Java Language

: 비트 시프트(Shift) 연산자

연산자	연산자의 기능	결합방향
<code><<</code>	<ul style="list-style-type: none">피연산자의 비트 열을 원쪽으로 이동이동에 따른 빈 공간은 0으로 채움예) <code>n << 2;</code> → n의 비트 열을 두 칸 원쪽으로 이동 시킨 결과 반환	→
<code>>></code>	<ul style="list-style-type: none">피연산자의 비트 열을 오른쪽으로 이동이동에 따른 빈 공간은 음수의 경우 1, 양수의 경우 0으로 채움예) <code>n >> 2;</code> → n의 비트 열을 두 칸 오른쪽으로 이동 시킨 결과 반환	→
<code>>>></code>	<ul style="list-style-type: none">피연산자의 비트 열을 오른쪽으로 이동이동에 따른 빈 공간은 0으로 채움예) <code>n >>> 2;</code> → n의 비트 열을 두 칸 오른쪽으로 이동 시킨 결과 반환	→

✓ 비트연산의 특징

- 원쪽으로의 비트 열 이동은 2의 배수의 곱
- 오른쪽으로의 비트 열 이동은 2의 배수의 나눗셈

- 정수 2 → 00000010 → 정수 2
- $2 \ll 1 \rightarrow 00000100 \rightarrow$ 정수 4
- $2 \ll 2 \rightarrow 00001000 \rightarrow$ 정수 8
- $2 \ll 3 \rightarrow 00010000 \rightarrow$ 정수 16

Java Language

: 3항 연산자(Conditional Operator)

- ▶ 세 개의 피연산자를 필요로 하는 연산자
- ▶ 조건식에 따라 **true**면 : 앞쪽의 연산식을, **false**면 : 뒤쪽의 피연산자를 선택한다

조건식 ? 값 또는 연산식 : 값 또는 연산식

조건식이 **true** 일 때

조건식이 **false** 일 때

Java Programming

Input and Output

콘솔 입출력

Java Input and Output

: 콘솔 출력

- ▶ System.out.println 메서드 : 출력 후 개행을 함
- ▶ System.out.print 메서드 : 출력 후 개행을 하지 않음

```
System.out.print("안녕");
System.out.println("하세요"); //안녕하세요
```

```
System.out.println("안녕하세요"); //안녕하세요
```

- ▶ 이스케이프 시퀀스 : 문자열 내 특별한 의미로 해석되는 문자 (\로 시작)

\n	개행
\t	탭(Tab)
\"	큰 따옴표(Quotation mark)
\\	역슬래쉬(Backslash)

Java Input and Output

: 콘솔 입력 - Scanner

- ▶ Scanner 클래스 : 다양한 리소스를 대상으로 입력을 받을 수 있도록 정의된 클래스

```
import java.util.Scanner;  
// ... 중략  
Scanner scanner = new Scanner(System.in);  
int value = scanner.nextInt();  
System.out.println(value);  
scanner.close();
```

✓ Scanner 클래스생성자

```
Scanner(File source)  
Scanner(InputStream source)  
Scanner(String source)  
Scanner(System.in)
```

입력받을 값에 맞는 메서드 선택

- public boolean nextBoolean()
- public byte nextByte()
- public short nextShort()
- public int nextInt()
- public long nextLong()
- public float nextFloat()
- public double nextDouble()
- public String nextLine()

Scanner 클래스는 단순히 키보드의 입력만을 목적으로 디자인된 클래스가 아니다.

연습문제

: 콘솔 입출력

[문제]

이름을 입력받아 출력하는 프로그램을 작성하세요

[문제]

이름과 나이를 입력받아 출력하는 프로그램을
작성하세요

```
<terminated> Console01 [Java Application] /Library/Java/JavaVi
이름을 입력해 주세요
이름:홍길동
당신의 이름은 홍길동입니다.
```

```
<terminated> Console01 [Java Application] /Library/Java/JavaVi
이름을 입력해 주세요
이름:홍길동
나이:23
당신의 이름은 홍길동, 나이는 23입니다.
```

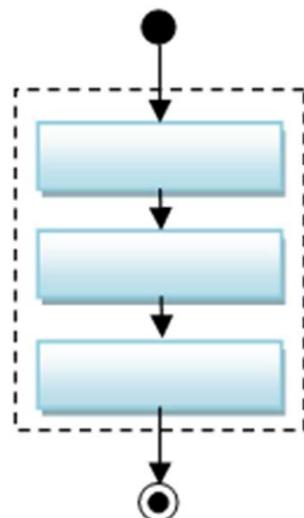
Java Programming

Java Conditionals

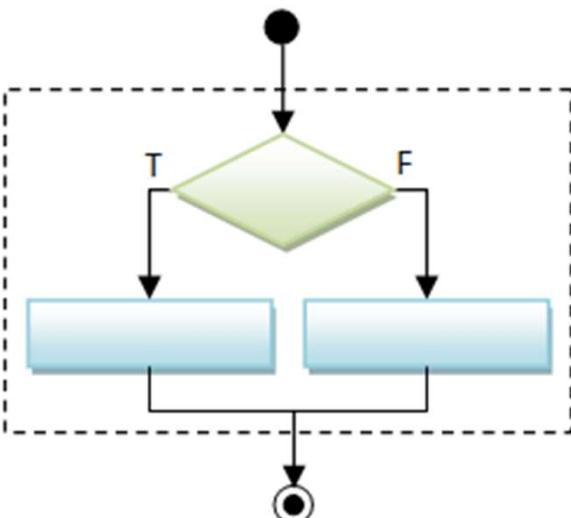
조건문

Flow Control

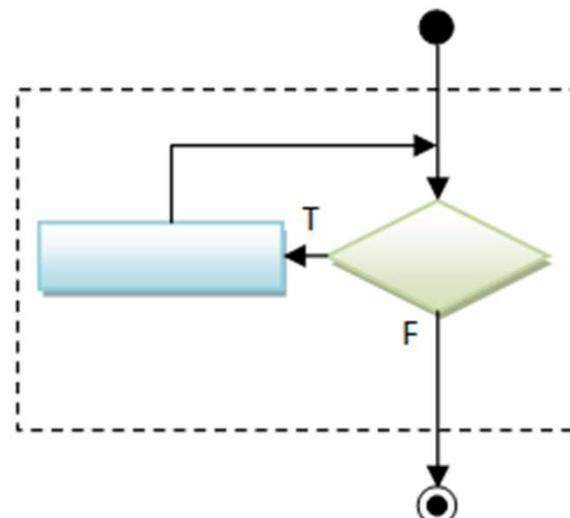
: Sequential vs Conditional vs Loop



Sequential



Conditional (Decision)



Loop (Iteration)

Java Conditionals

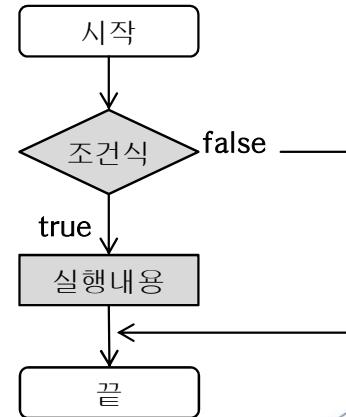
: if 조건문

```
if ( 조건식 ) {  
    /* 조건식이 true 이면 실행되는 영역 */  
}
```

어떤 조건에 의해 흐름을 결정하는 것이어서
Decision Making이라 부르기도 한다

[예제]

점수를 입력받아
점수가 60점 이상이면 "합격입니다." 를 출력하세요



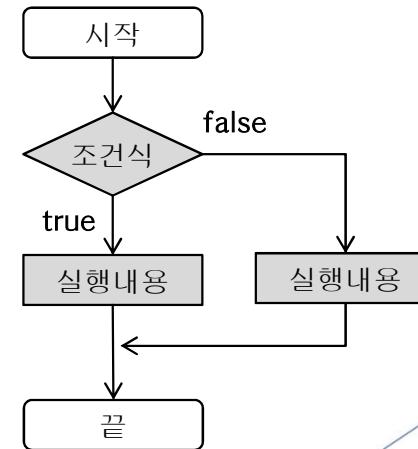
Java Conditionals

: if ~ else 조건문

```
if ( 조건식 ) {  
    /* 조건식이 true 이면 실행되는 영역 */  
} else {  
    /* 조건식이 false 이면 실행되는 영역 */  
}
```

[예제]

점수를 입력 받아
60점 이상이면 “합격입니다.”
60점 미만이면 “불합격입니다.”
를 출력하세요



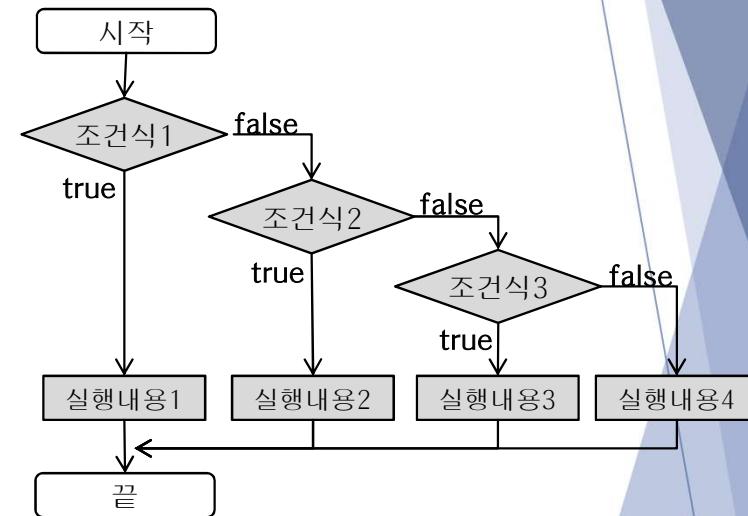
Java Conditionals

: if ~ else if ~ else 조건문

```
if ( 조건식1 ) {  
    /* 조건식1 이 true 이면 실행되는 영역 */  
} else if (조건식2) {  
    /* 조건식2 가 true 이면 실행되는 영역 */  
} else if (조건식3) {  
    /* 조건식3 이 true 이면 실행되는 영역 */  
} else {  
    /* 위의 조건이 모두 해당되지 않으면 실행되는 영역 */  
}
```

[예제]

숫자를 입력받아
수가 0보다 크면 “양수” 영보다 작으면 “음수” 0일때는 “0입니다.”을 출력하세요



연습문제

: 조건문 연습

[문제]

숫자를 입력받아 아래와 같이 출력되는
프로그램을 작성하세요

입력받은 수가

양수일때

짝수이면 “짝수” 출력

홀수 이면 “홀수” 출력

음수이면 “음수” 라고 출력

0 이면 “0” 으로 출력



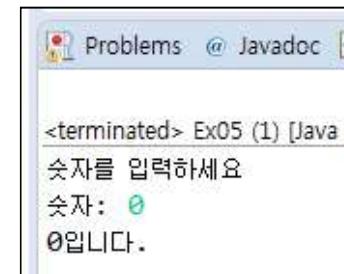
```
<terminated> Ex05 (1) [Java Application]
숫자를 입력하세요
숫자: 8
짝수입니다.
```



```
<terminated> Ex05 (1) [Java Application]
숫자를 입력하세요
숫자: 17
홀수입니다.
```



```
<terminated> Ex05 (1) [Java Application]
숫자를 입력하세요
숫자: -20
음수입니다.
```



```
<terminated> Ex05 (1) [Java Application]
숫자를 입력하세요
숫자: 0
0입니다.
```

연습문제

: if ~ else if ~ else 연습

[문제]

과목번호를 입력받아 강의실 번호를 출력하는
프로그램을 작성하세요

과목 code값이

1이면 "R101호 입니다."

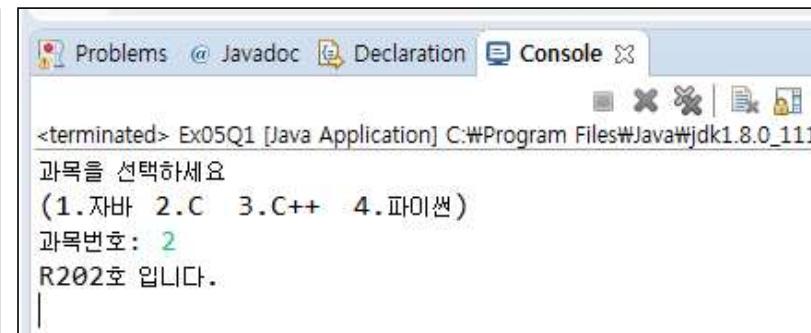
2이면 "R202호 입니다."

3이면 "R303호 입니다."

4이면 "R404호 입니다."

나머지는 "상담원에게 문의하세요"

를 출력하세요

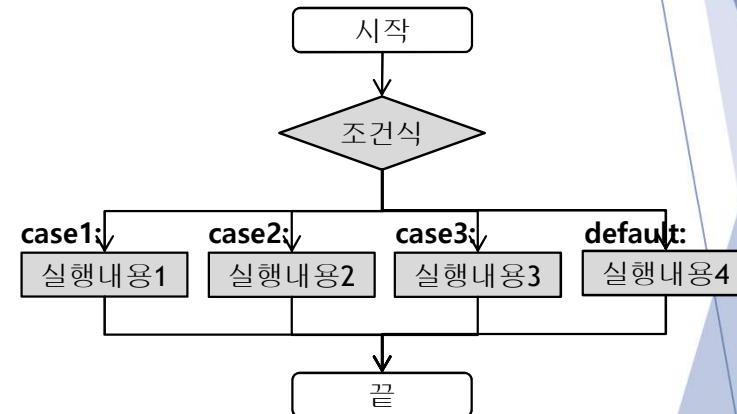


The screenshot shows a Java application window with tabs for Problems, Javadoc, Declaration, and Console. The Console tab is active, displaying the following text:
<terminated> Ex05Q1 [Java Application] C:\Program Files\Java\jdk1.8.0_111\
과목을 선택하세요
(1.자바 2.C 3.C++ 4.파이썬)
과목번호: 2
R202호 입니다.

Java Conditionals

: switch ~ case 조건문

```
switch (변수) {  
    case 값1 :  
        /* 변수 값이 값1일 때 실행내용*/  
        break;  
    case 값2 :  
        /* 변수 값이 값2일 때 실행내용*/  
        break;  
    case 값3 :  
        /* 변수 값이 값3일 때 실행내용*/  
        break;  
    default :  
        /* 해당 내용이 없을 때 실행내용*/  
        break;  
}
```



case의 값에는 수치형 뿐 아니라, char, String 도 올 수 있다

연습문제

: switch ~ case 연습

- ▶ if 연습문제와 같은 문제지만 switch ~ case를 이용하여 풀어 봅시다

[문제]

과목번호를 입력받아 강의실 번호를 출력하는 프로그램을 작성하세요

과목 code값이

1이면 "R101호 입니다."

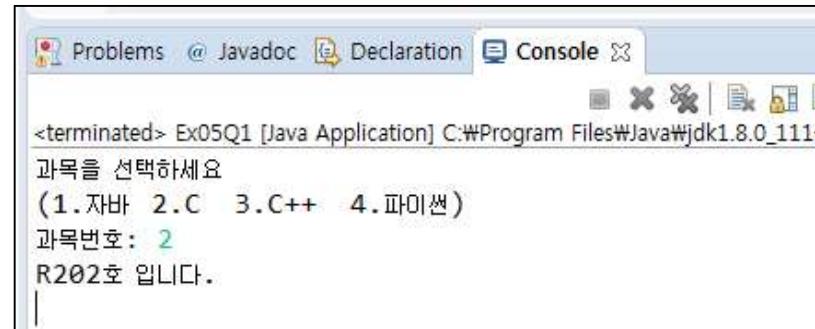
2이면 "R202호 입니다."

3이면 "R303호 입니다."

4이면 "R404호 입니다."

나머지는 "상담원에게 문의하세요"

를 출력하세요



The screenshot shows a Java application window with the title bar 'Ex05Q1 [Java Application] C:\Program Files\Java\jdk1.8.0_111\'. The 'Console' tab is selected. The output window displays the following text:
<terminated>
과목을 선택하세요
(1.자바 2.C 3.C++ 4.파이썬)
과목번호: 2
R202호 입니다.

if-else 문의 조건식 부분이 모두 == 로 표현될 때 switch문을 사용

연습문제

: switch ~ case 연습

[문제]

월을 입력받아 해당월의 일수를 출력하는
프로그램을 작성하세요

1월이면 "31일"
2월이면 "28일"
3월이면 "31일"
4월이면 "30일"
5월이면 "31일"
6월이면 "30일"
7월이면 "31일"
8월이면 "31일"
9월이면 "30일"
10월이면 "31일"
11월이면 "30일"
12월이면 "31일"

The image shows three separate instances of a Java application's console window, each displaying a different month input and its corresponding day count. The windows are arranged vertically.

- Top Window:** Displays the message "월을 입력하세요" followed by the number "2" and the response "28일 입니다."
- Middle Window:** Displays the message "월을 입력하세요" followed by the number "9" and the response "30일 입니다."
- Bottom Window:** Displays the message "월을 입력하세요" followed by the number "5" and the response "31일 입니다."

Each window has tabs for Problems, Javadoc, Declaration, and Console, with the Console tab being active. The title bar of each window indicates it is a terminated Java Application named Ex06.

연습문제

: 조건문 연습

[문제]

점수를 입력받아 입력된 수가 3의 배수인지 판별하는 프로그램을 작성하세요

```
<terminated> Ex01 (3) [Java Application] C:\Program Files\Java\jdk1.8.0_111\
점수를 입력하세요
15
3의 배수입니다.
```

[문제]

점수를 입력받아 등급을 표시하는 프로그램을 작성하세요

90점 이상 이면 "A등급"
80점 이상~89점 이면 "B등급"
70점 이상~79점 이면 "C등급"
60점 이상~69점 이면 "D등급"
60점 미만이면 "F등급"

```
<terminated> Ex01 (3) [Java Application] C:\Program Files\Java\jdk1.8.0_111\
숫자를 입력하세요
55
55은 3의 배수가 아닙니다.
```

```
<terminated> Ex07Q [Java Application] C:\Program Files\Java\jdk1.8.0_111\
점수를 입력하세요 : 94
A등급
```

```
<terminated> Ex07Q [Java Application] C:\Program Files\Java\jdk1.8.0_111\
점수를 입력하세요 : 78
C등급
```

if문과 switch문 비교

```
switch (month) {  
    case 2:  
        days = 28;  
        break;  
  
    case 4:  
    case 6:  
    case 9:  
    case 11:  
        days = 30;  
        break;  
  
    default:  
        days = 31;  
        break;  
}
```

```
if (month == 2) {  
    days = 28;  
} else if ((month == 4) || (month == 6) || (month == 9)  
|| (month == 11)) {  
    days = 30;  
} else {  
    days = 31;  
}
```

break 없는 switch-case문 짐작하지 않음

Java Programming

Java Loop
반복문

Java Loop

: while 반복문

▶ 기본 구문 : `while (condition) { statements }`

- ▶ condition 값이 참인 동안 내부 블록을 반복하여 실행함
- ▶ 초기 condition 값이 false 이면 내부 블록은 실행되지 않음
 - ▶ Condition 값이 계속 true 면 무한 반복
 - ▶ 무한 반복은 주의해야 하지만, 의도적으로 무한 반복을 실행하는 경우도 있음

```
// 1에서 10까지 숫자를 출력함
int x = 1;
while (x <= 10){
    System.out.println(x);
    x = x + 1;
```

[예제]

다음과 같이 출력되는 프로그램을 작성하세요



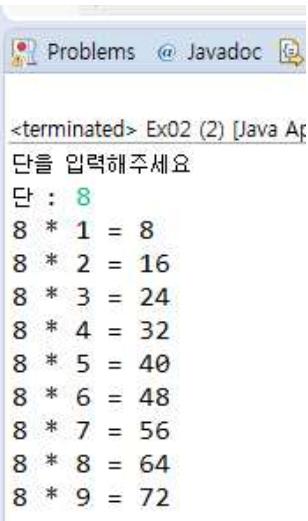
```
I like Java0
I like Java1
I like Java2
I like Java3
I like Java4
```

연습문제

: while 반복문 구구단

[문제]

숫자를 입력받아 입력한 숫자(단)의
구구단을 출력하세요



The screenshot shows a Java application window titled "Problems @ Javadoc". The console output shows the program has terminated. It prompts the user to input a number ("단을 입력해주세요") and then prints the multiplication table for that number. The user input is "8", and the output is:

```
단 : 8
8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
```

```
public class Ex02 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int dan ;
        int i = 1;

        //메세지출력, dan값 입력

        while(//조건문){
            //구구단 출력 코드
        }
        sc.close();
    }
}
```

Java Loop

: do ~ while 반복문

- ▶ do 블록을 우선 한번 수행하고 반복 여부를 while에서 확인
 - ▶ 적어도 한 번은 수행되어야 하는 반복 구문에 사용
 - ▶ while 문에서 확인할 조건이 반복 구문 내에서 할당되는 경우

[문제]

사용자의 숫자를 입력받아 더하는 프로그램을 작성하세요(0이면 종료)

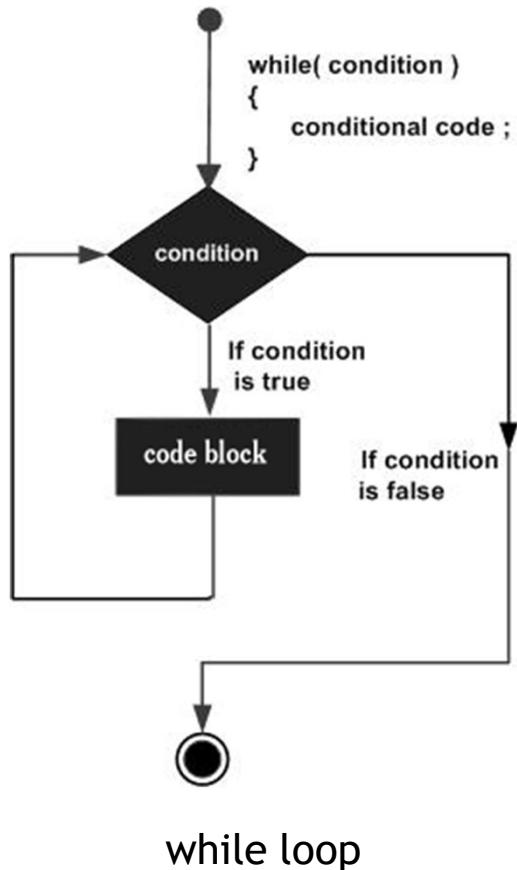
```
Problems @ Javadoc Declarations  
<terminated> Ex07 (1) [Java Application]   
숫자를 입력하세요. [0이면 종료]  
4  
합계: 4  
5  
합계: 9  
6  
합계: 15  
0  
합계: 15  
종료
```

```
Problems @ Javadoc Declarations  
<terminated> Ex07 (1) [Java Application]   
숫자를 입력하세요. [0이면 종료]  
0  
합계: 0  
종료
```

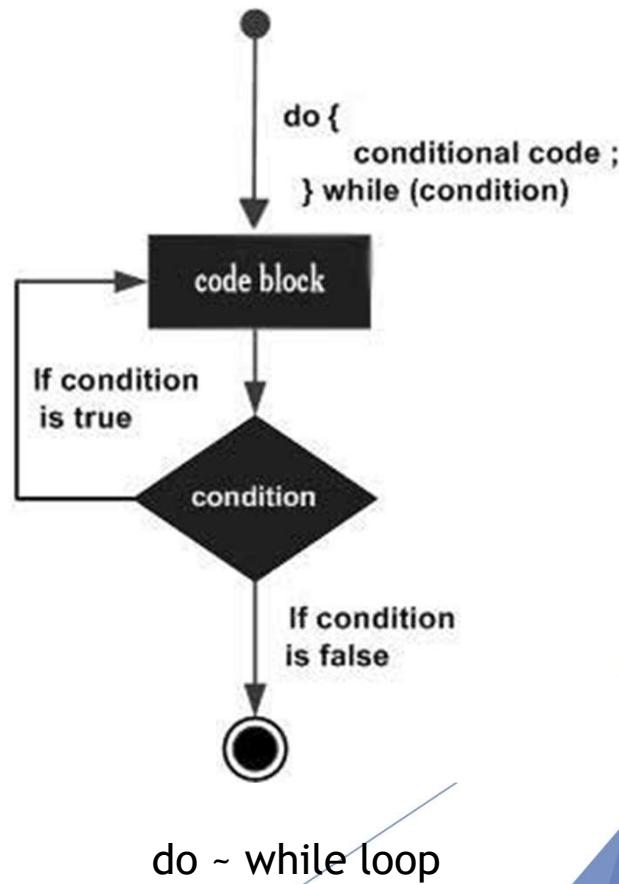
```
int total = 0;  
int value = 0;  
do {  
    System.out.print("숫자입력[ 0 to quit ] >");  
    value = scanner.nextInt();  
    total += value;  
} while(value != 0);
```

Java Loop

: while vs do ~ while



루프를 빠져나가는 것은 오롯이
개발자의 몫!
의도한 것이 아니라면 무한루프는 피하도록 한다

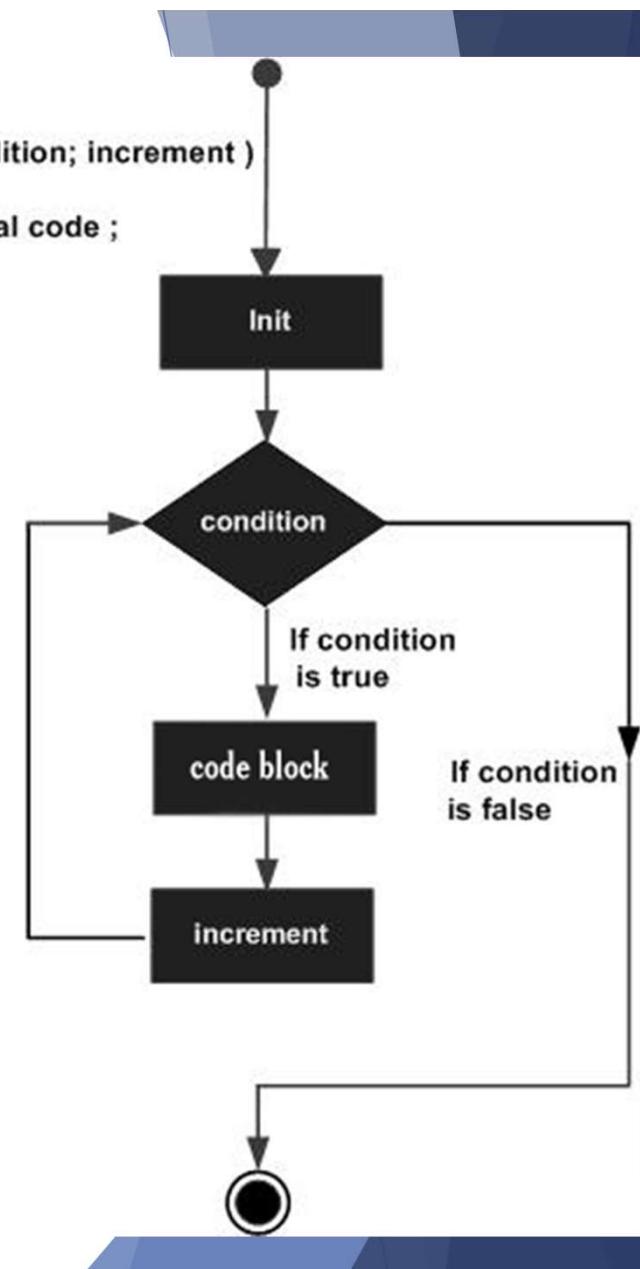


Java Loop

: for 반복문

```
for ( 초기화; 조건검사; 증감연산 ) {  
    statement1;  
    statement2;  
    .....  
}  
  
for ( int i=0; i<10; i++ ) {  
    a = a + 1;  
}
```

```
for( init; condition; increment )  
{  
    conditional code ;  
}
```



Java Loop

: for vs while

```
int num=0; ①  
while( num<5 ) ②  
{  
    System.out.println( num );  
    num++; ③  
}
```

```
① for( int num=0 ; num<5 ; num++ )  
{  
    System.out.println( num );  
}
```

```
for( 초기화; 조건식; 증감식 ){  
    조건식이 맞을때 실행내용  
}
```

① → 반복의 횟수를 세기 위한 변수 (**초기화**)

② → 반복의 조건 (**조건식**)

③ → 반복의 조건을 무너뜨리기 위한 연산(**증감식 : 탈출조건**)

Java Loop

: for 반복문 실행 흐름

첫 번째 루프의 흐름

1 → 2 → 3 → 4 [i=1]

두 번째 루프의 흐름

2 → 3 → 4 [i=2]

세 번째 루프의 흐름

2 → 3 → 4 [i=3]

네 번째 루프의 흐름

2 [i=3] 따라서 탈출 !

```
for( 1. int i=0 ; 2. i<3 ; 3. i++ )  
{ 4. System.out.println("...");  
}
```

I like Java 0

I like Java 1

I like Java 2

연습문제

: for 반복문 구구단

[문제]

숫자를 입력받아 입력한 숫자(단)의
구구단을 출력하세요 (for문으로 작성)

```
<terminated> Ex02 (2) [Java Ap
단 : 8
8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
```

```
public class Ex02 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int dan ;
        int i = 1;

        //메시지 출력, 단 입력

        for(//조건문){
            //구구단 출력 코드
        }
        sc.close();
    }
}
```

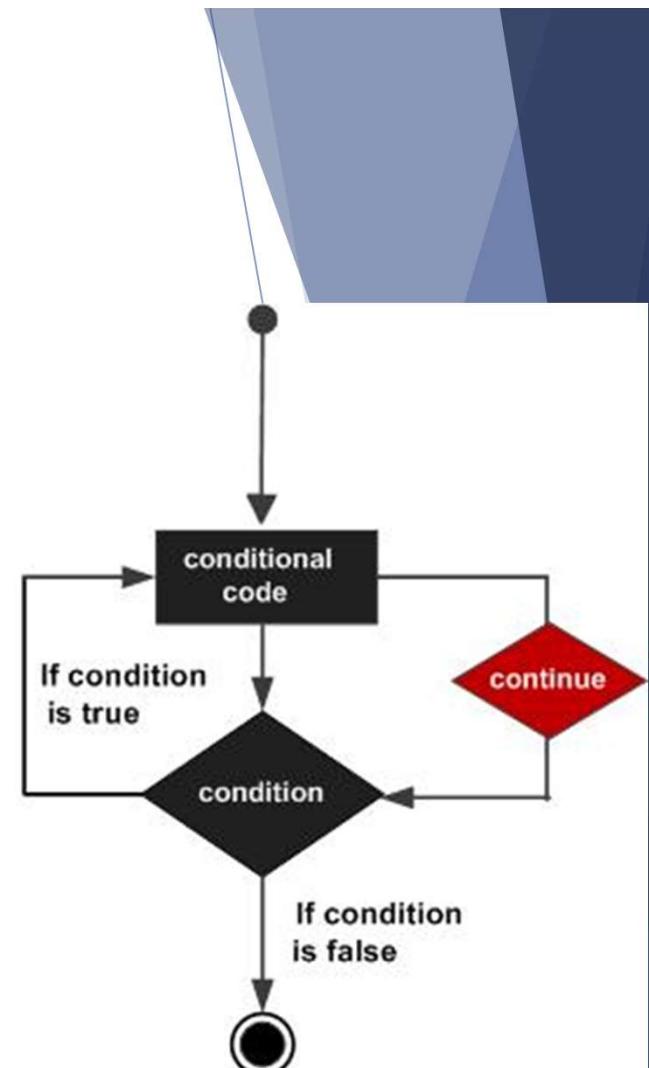
Java Loop

: continue

▶ 흐름 제어 : continue

- ▶ 반복문 내에서 continue를 만나면 이후 문장은 수행하지 않음
- ▶ 반복 블록을 벗어나지 않고 블록의 가장 마지막으로 이동, 다시 반복 조건을 검사

```
for ( int i = 0; i < 20; i++ ) {  
    if( i % 2 == 0 || i % 3 == 0 ) {  
        continue;  
    }  
  
    System.out.println( i );  
}
```



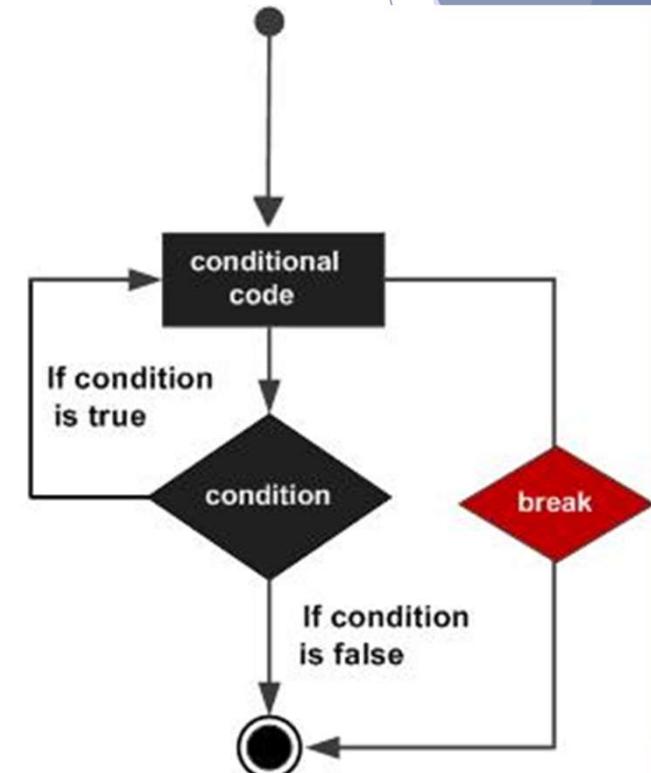
Java Loop

: break

▶ 흐름 제어 : break

- ▶ 반복문 수행을 중단하고 반복 블록 다음 문장을 실행
- ▶ 중첩된 반복문에서 한 단계씩 반복문을 벗어남

```
while( true ) {  
    sum += num;  
  
    if( sum > 5000 ) {  
        break;  
        num++;  
    }  
}
```



Java Loop

: continue vs break

continue

```
while (true) {  
    //...  
    if (x < 0) {  
        continue;  
    }  
    //...  
}
```



break

```
while( true ) {  
    //...  
    if( x < 0 ) {  
        break;  
    }  
    //...  
}
```



루프 탈출

연습문제

: continue, break

[문제]

1에서 20까지의 수에서 2의배수와 3의배수를 제외한 숫자를 출력하세요
(for문, continue문 사용)



```
Problems Javadoc Declaration Correspondence
<terminated> Multiple57 [Java Application] C:\WPro...
1
5
7
11
13
17
19
```

연습문제

: 반복문 연습

- ▶ **for** 루프도 좋고, **while** 루프도 좋습니다. 자신있는 루프 방식으로 문제들을 구현해 봅시다

[문제]

다음과 같이 출력되는 구구단을 출력하세요

```
<terminated> Ex06 (1) [Java Application] C:\#Program Files\Java\jdk1.8
2*1 = 2
2*2 = 4
2*3 = 6
[중간생략]
9*7 = 63
9*8 = 72
9*9 = 81
```

[문제]

아래와 같이 출력되는 프로그램을 작성하세요

```
<terminated> Ex08 [Java Application] C:\#Program Files\Java\jdk1.8
*
**
***
****
*****
******
*****
```

생각해 봅시다

: for vs while

- ▶ 아래와 같은 문제가 있다고 가정할 때, for 문이 좋을까요? while 문이 좋을까요?

[문제]

6의 배수이자 14의 배수인 가장 적은 정수 찾기



Hint: 반복의 횟수를 알 수 있을 때 = for
반복의 횟수를 알 수 없을 때 = while

Random

: Math.random()으로 정수값 생성하기

```
int num = (int)(Math.random() * 최대값) + 최소값;
```

예) 주사위

```
int num = (int)(Math.random()*6)+1
```

- 0.0 이상 1.0 미만인 실수값 반환 (1.0은 포함되지 않는다)
- double 형 값 반환
- double 형을 int 형으로 강제 형변환 하면 소수점 아래가 버림 처리 된다.
4.175353107765874 → 4
- 정수 난수값 필요시 주로 사용

연습문제

: Math.random()을 이용한 미니 로또

[문제]

1~45 까지의 숫자중 임의의 6개의 숫자를 출력하세요
(중복체크는 하지 말 것)



Problems @ Javadoc Declaration
<terminated> MiniLotto [Java Application] C
6 40 20 39 27 25



Problems @ Javadoc Declaration
<terminated> MiniLotto [Java Application]
15 15 23 29 43 26

for 혹은 while 문을 이용하여 1~45사이의 정수값 여섯 개를 생성한다

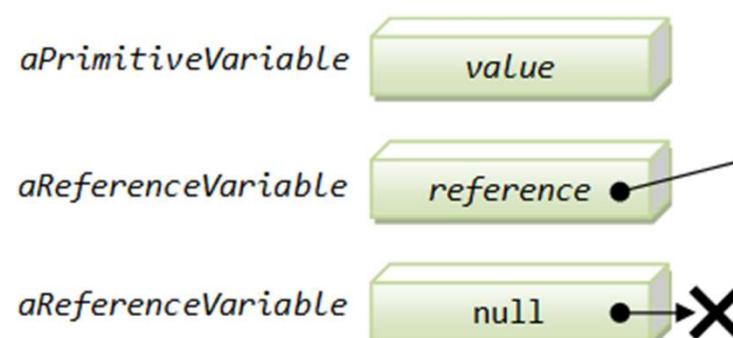
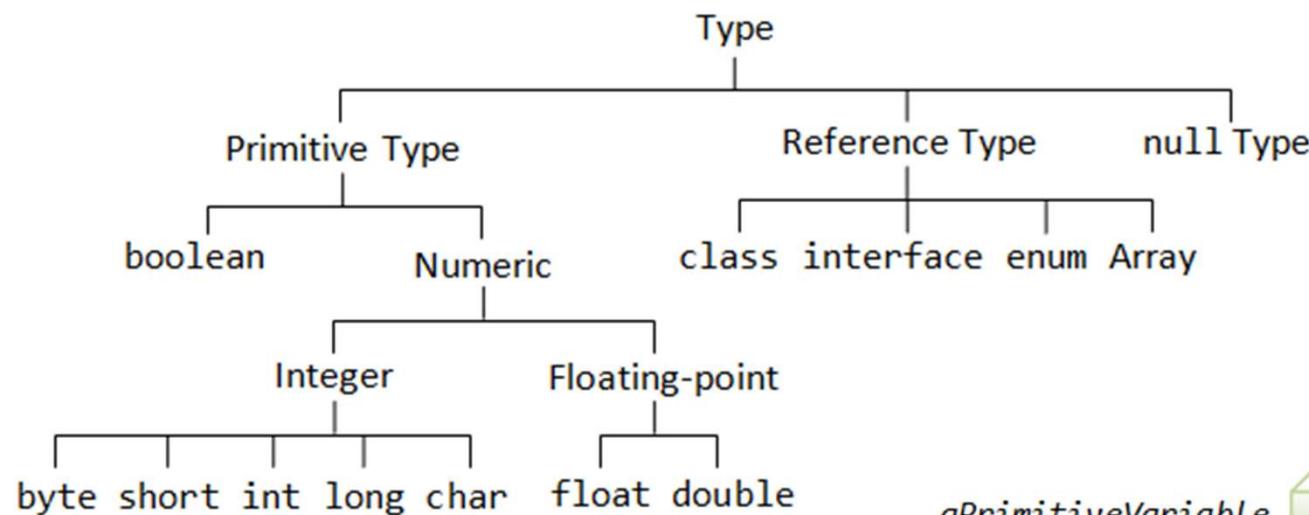
Java Programming

Reference Type

참조 타입

Reference Type

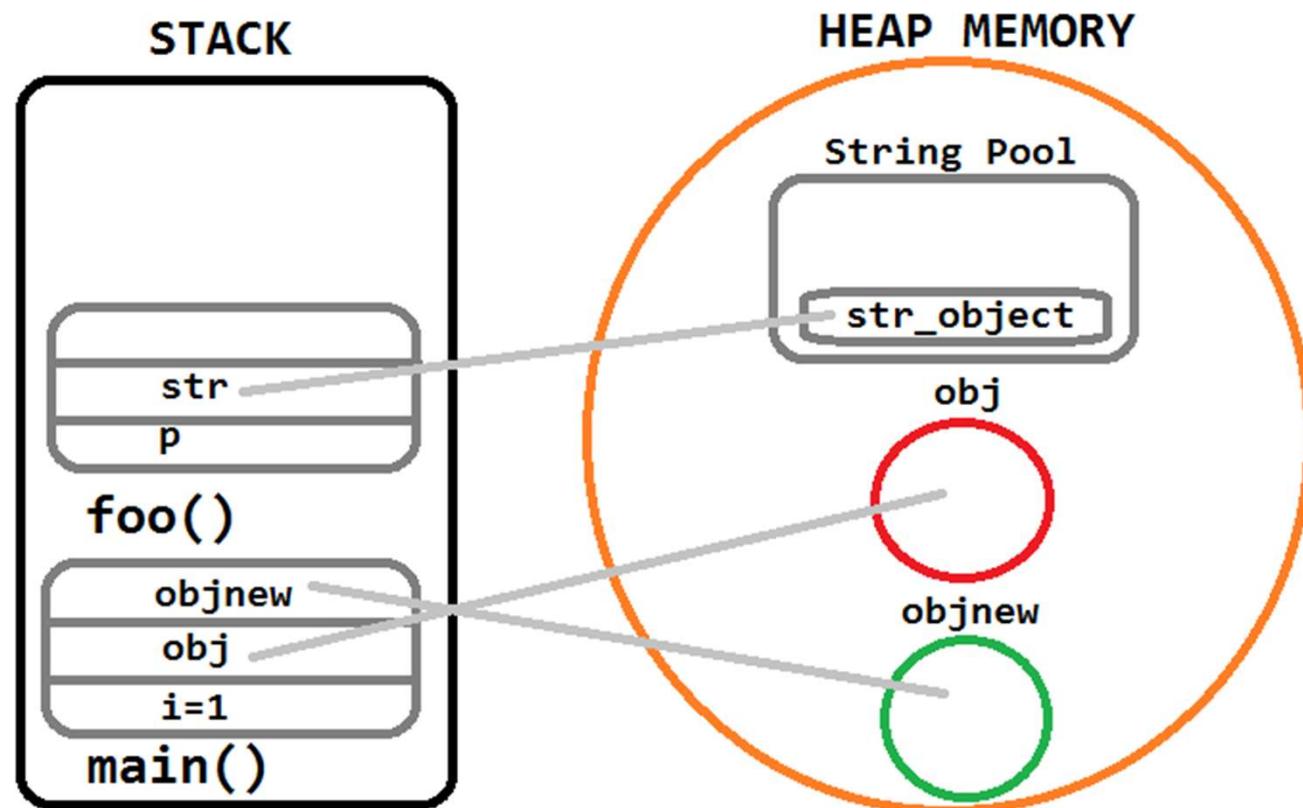
: Primitive Type vs Reference Type



Heap

Java의 메모리 사용 구조

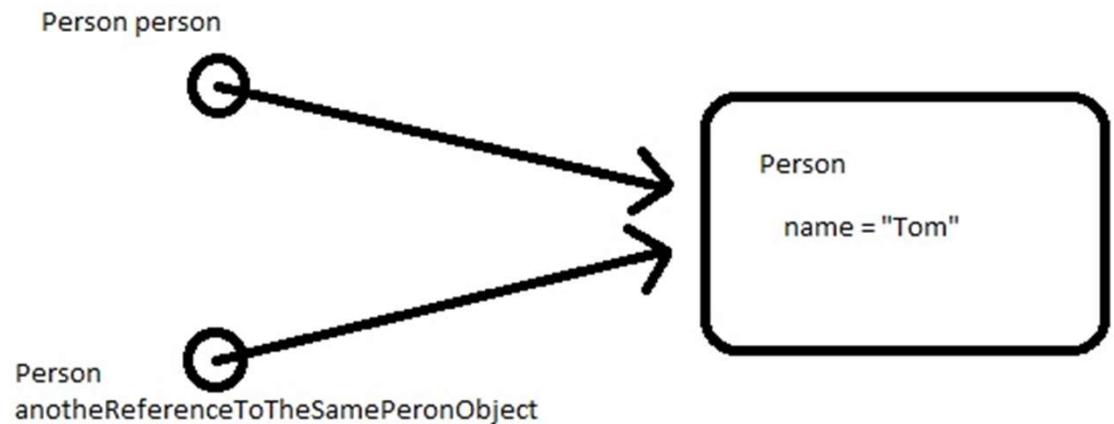
: Stack과 Heap 영역



Reference Type

: 참조 변수의 ==, !=

- ▶ 참조 변수의 ==, != 는 값이 같은지 비교하는 것이 아니라 동일한 객체를 참조하는지 아닌지를 확인하기 위해 사용
 - ▶ refVal1 == refVal2 : refVal1과 refVal2가 같은 객체를 참조하는가 확인
 - ▶ refVal1 != refVal2 : refVal1과 refVal2가 다른 객체를 참조하는가 확인
- ▶ null : 참조하는 객체가 없음을 나타냄
 - ▶ null인 참조 변수를 코드상에서 사용하고자 하면 NullPointerException 발생



Reference Type

: String

- ▶ 문자열을 저장하기 위해서는 **String** 타입으로 선언

- ▶ `String varName; // 선언`

- ▶ 리터럴: 소스코드상에 고정된 값

- ▶ `varName = "문자열"; // 문자열 할당. "" 사이에 문자열 리터럴 넣음`

- ▶ `String varName = "문자열"; // 선언과 할당을 동시에`

- ▶ `String varName = new String("문자열");`

Reference Type

: String 값 비교

- ▶ String 값이 같은지 비교하려면 .equals 메서드를 사용한다
 - ▶ `strVal1.equals(strVal2); // strVal1과 strVal2가 같은 값을 가지고 있는가?`

Reference Type

: String의 포맷 - printf

Examples

```
System.out.printf("Hello %s of %s%n", "World", "Java");
String s = String.format("The meaning of the %s is %d", "universe", 42);
```

Format	Description	Format	Description	Format	Description
%b	Boolean data	%c	Charactor	%d	Decimal
%e	Big Deciaml	%f	Float-point	%x	Hex-Decimal
%o	As Octal	%s	String	%n	New Line
%t	Date/time	%%	Percent		

문자열을 보다 풍부하고 편리하게 표현하기 위해 .format 메서드를 이용한다

열거형(enum)

- ▶ 몇 가지로 한정된 값만을 가지는 경우
- ▶ 예) 요일, 계절, 성별 등
- ▶ public enum 열거타입명 { 열거 상수, ... }
- ▶ 열거 상수는 모두 대문자로 작성(관례)
- ▶ 여러 단어로 구성될 경우 _로 연결(관례)
 - ▶ 예) RESULT_SUCCESS, RESULT_FAILED

```
public enum DayOfWeek {  
    SUNDAY,  
    MONDAY,  
    TUESDAY,  
    WEDNESDAY,  
    THURSDAY,  
    FRIDAY,  
    SATURDAY;  
}
```

열거 타입명

열거 상수

열거 객체의 메서드

Return Type	메서드	설명
String	<code>name()</code>	열거 타입의 문자열을 리턴
int	<code>ordinal()</code>	열거 객체의 순번을 리턴
int	<code>compareTo(열거객체)</code>	열거 객체를 비교, 순번차를 리턴
열거 타입	<code>valueOf(String name)</code>	주어진 문자열의 열거 객체를 리턴
열거 타입	<code>values()</code>	모든 열거 객체들을 배열로 리턴

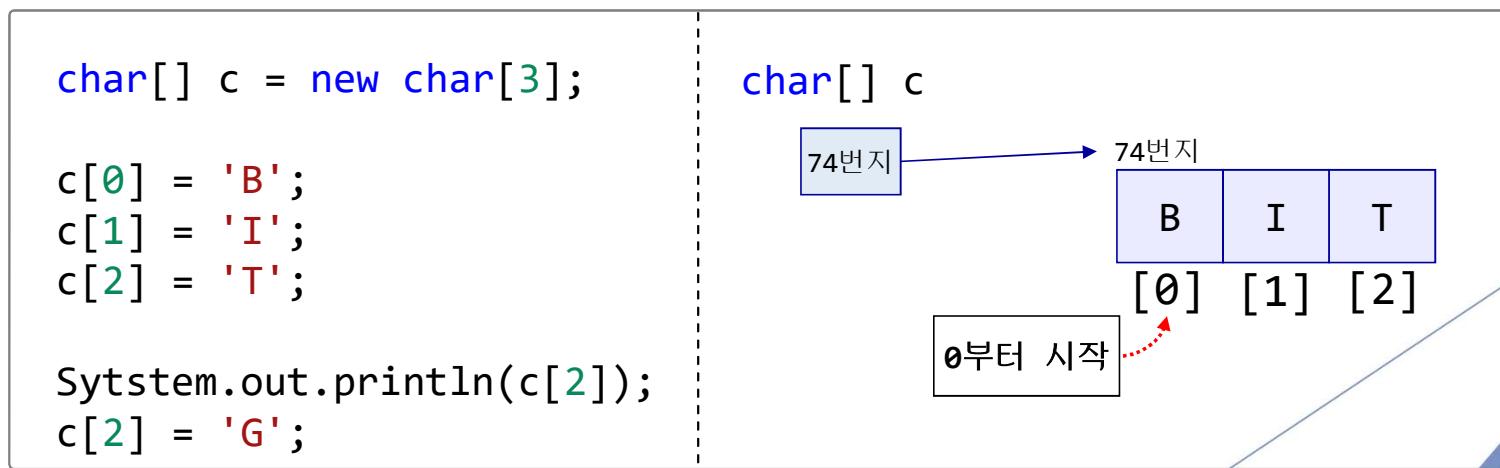
Reference Type

: Array (배열)

30명의 성적을 처리한다고 생각해보자
30개의 변수를 선언할 것인가?

▶ 배열이란?

- ▶ 동일한 자료 유형의 여러 값들로 이루어진 객체(Object)
- ▶ new로 생성되는 참조 자료형
- ▶ 배열에 포함된 값들은 기본 자료형(Primitive Type)일 수도 있고 다른 객체를 참조하는 참조 자료형(Reference Type)일 수도 있음



Reference Type

: Array (배열)의 선언

- ▶ 배열을 선언하는 두 가지 방법
 - ▶ 타입[] 변수;
 - ▶ 타입 변수[];

```
int[] intArray; // = int intArray[];  
double[] doubleArray; // = double doubleArray[];  
String[] stringArray; // = String stringArray[];
```

Reference Type

: Array (배열)의 초기화

- ▶ 값 목록이 있다면, {}으로 값의 목록을 지정하여 초기화할 수 있음
- ▶ 타입 배열명[] = { 값0, 값1, 값2, 값3, ...};

```
String[] days = {"월요일", "화요일", "수요일", "목요일", "금요일", "토요일", "일요일"};
```

- ▶ 이 방식은 배열 변수 선언과 동시에 해 주어야 허용된다.
- ▶ 다음과 같은 방식으로 new 연산자 뒤에 나열할 수도 있다.

```
String[] days;  
days = new String[] {"월요일", "화요일", "수요일", "목요일", "금요일", "토요일", "일요일"};
```

Reference Type

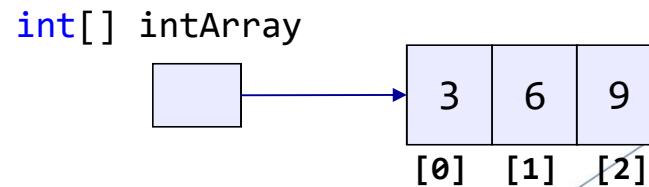
: Array (배열)의 초기화

- ▶ new 연산자로 배열 생성

```
int[] intArray = new int[3];  
intArray[0] = 3;  
intArray[1] = 6;  
intArray[2] = 9;
```

주의: 타입 선언시 배열 크기 지정 안됨

```
int[5] intArray = new int[]; // (x)
```

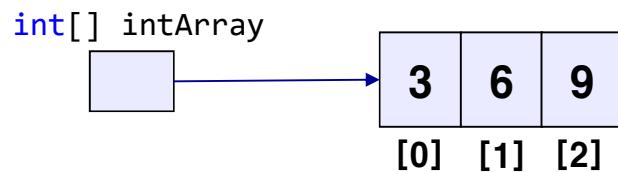


Reference Type

: Array (배열)의 사용

```
int[] intArray = new int[3]  
  
intArray[0] = 3;  
intArray[1] = 6;  
intArray[2] = 9;
```

```
System.out.println(intArray[0]);  
System.out.println(intArray[1]);  
System.out.println(intArray[2]);
```



```
for(int i=0; i<3; i++){  
    System.out.println(intArray[i]);  
}
```

```
for(int i=0; i<intArray.length; i++){  
    System.out.println(intArray[i]);  
}
```

배열의 `length` 멤버는 배열의 길이를 알아낼 때 사용할 수 있다

Reference Type

: Array (배열)의 선언 → 생성 → 초기화

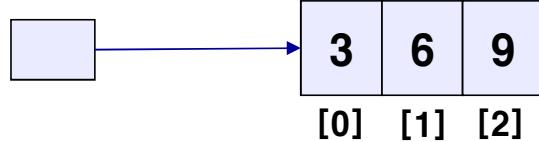
일반적인 인스턴스 배열의 선언과 생성

```
int[] intArray = new int[3];
```

배열의 초기화

```
intArray[0]=3;  
intArray[1]=6;  
intArray[2]=9;
```

int[] intArray



일반적인 인스턴스 배열의 선언과 생성

```
int[] intArray = new int[3];
```

생성과 동시에 초기화, 길이정보 생략 가능

```
int[] intArray= new int[3] {1,2,3}; (X)  
int[] intArray= new int[] {1,2,3};
```

줄여서 표현 가능

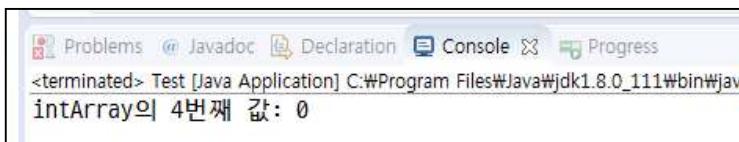
```
int[] intArray= {1,2,3};
```

Reference Type

: Array (배열) `ArrayIndexOutOfBoundsException`

- ▶ 배열의 범위를 벗어난 곳에 접근하면 `ArrayIndexOutOfBoundsException`을 일으킨다
- ▶ 배열의 인덱스는 0부터 시작한다는 점을 잊지 말자
 - ▶ 배열의 인덱스 범위는 0 ~ (`length - 1`)

```
int[] intArray;  
intArray = new int[5];  
intArray[0] = 3;  
intArray[1] = 6;  
intArray[2] = 9;  
  
int result = 0;  
for ( int i = 0; i <= intArray.length; i++ ){  
    result = result + intArray[i];  
}  
  
System.out.println("intArray의 4번째 값: " + intArray[3]);
```



Reference Type

: main 함수에서의 배열

배열 형태의 값 목록이 main 함수로 전달된다

```
package com.javaex.helloworld;

public class MainArgs {
    public static void main(String[] args) {
        for (int i = 0; i < args.length; i++) {
            System.out.println("args [" + i + "] : " + args[i]);
        }
    }
}
```

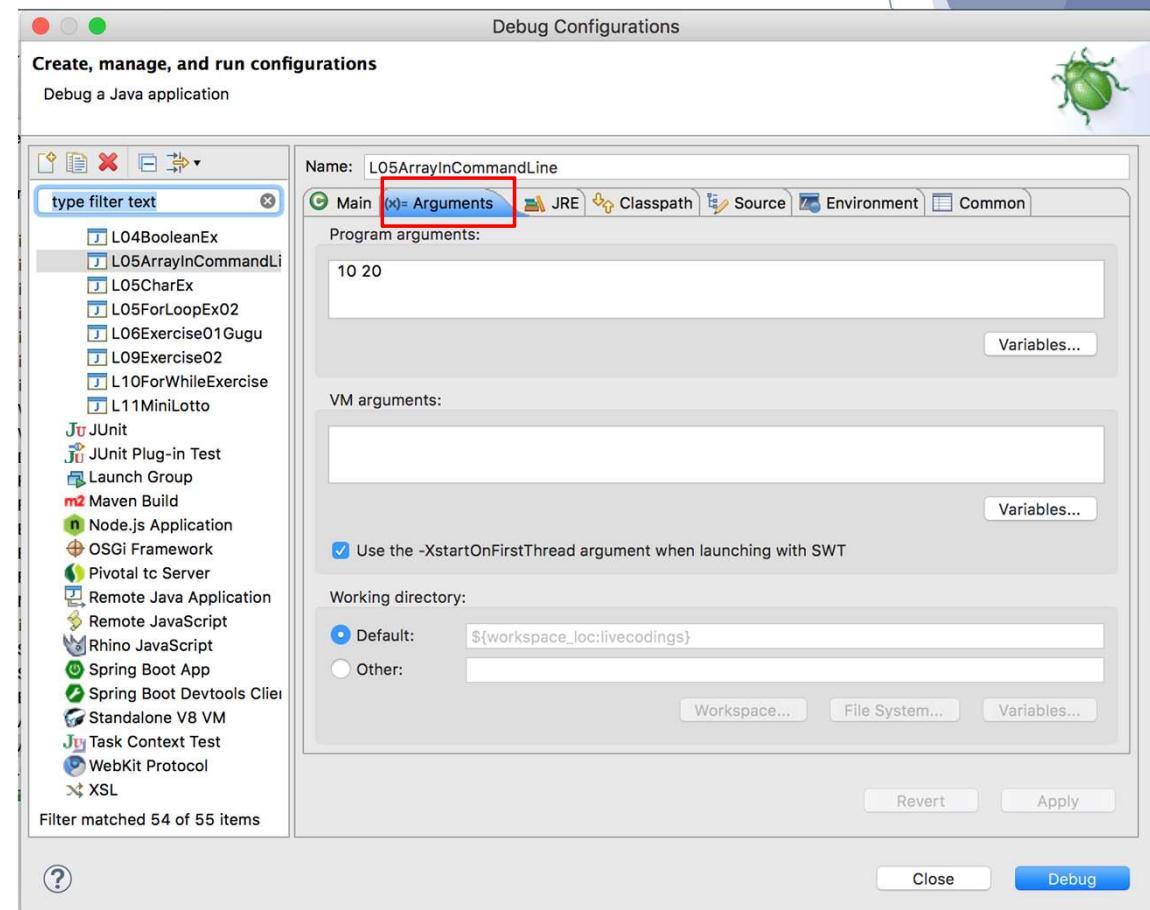
> java MainArgs Java Language is fun

```
args[0] = Java
args[1] = Language
args[2] = is
args[3] = fun
```

Reference Type

: Eclipse에서 main args 인수 전달

- ▶ Run > Run Configurations에서 Arguments 탭을 클릭하면 main 함수로 넘겨줄 인자를 정의할 수 있다

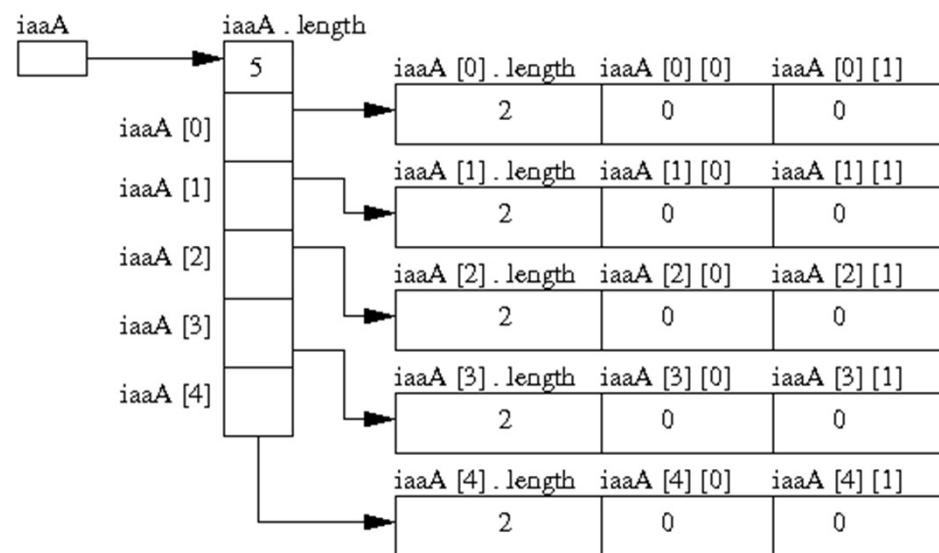


Reference Type

: 다차원 배열

```
int [][] iaaA = new int [5][2];
```

Layout of Memory



▶ 타입[][] 배열명 = new 타입[열_수][행_수];

```
int[][] twoDimens = int[5][2];
```

-> 5행 2열 배열 생성

3차원 이상 배열은 다루기 힘드니 조심해서 사용해야 한다

Reference Type

: 배열의 복사

- ▶ 배열은 한번 생성하면 크기를 변경할 수 없다
- ▶ 추가로 저장 공간이 필요하다면 원래 배열보다 큰 배열을 만들고 이전 배열의 항목값을 복사해야 한다

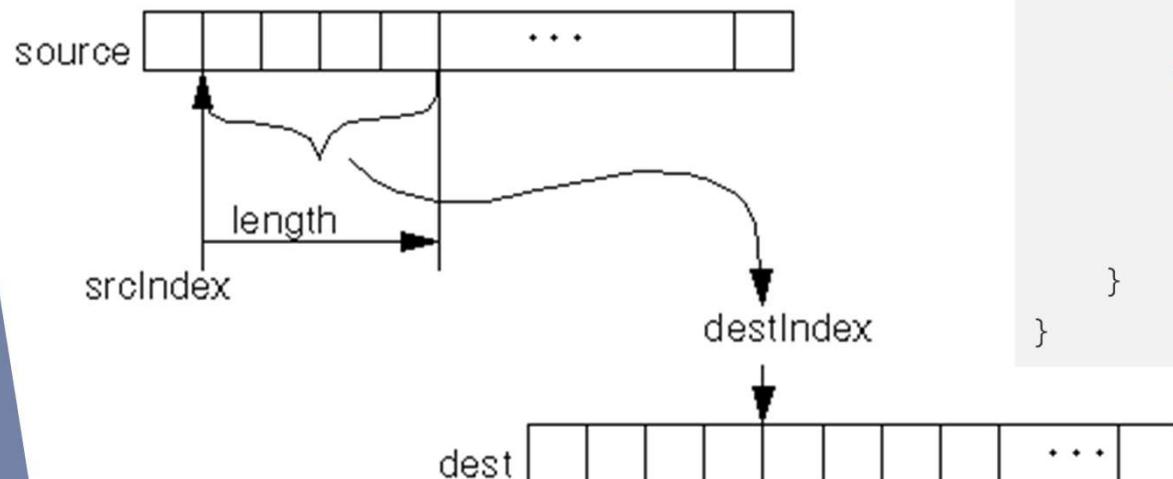
// For 문을 이용한 Copy

```
public class ArrayCopyByFor {  
    public static void main(String[] args) {  
        int[] intArray = { 1, 2, 3 };  
        int[] newArray = new int[10];  
        for (int i = 0; i < intArray.length; i++) {  
            newArray[i] = intArray[i];  
        }  
        for (int i = 0; i < newArray.length; i++) {  
            System.out.print(newArray[i] + " ");  
        }  
        System.out.println();  
    }  
}
```

Reference Type

: 배열의 복사

- ▶ System.arraycopy()로 배열을 복사할 수도 있다
- ▶ System.arraycopy(원본배열, 시작인덱스, 타겟배열, 시작인덱스, 복사할 길이)



```
// System.arraycopy()를 이용한 배열 복사
public class ArrayCopyByArrayCopy {
    public static void main(String[] args) {
        int[] intArray = { 1, 2, 3 };
        int[] newArray = new int[10];

        System.arraycopy(intArray, 0,
                        newArray, 0,
                        intArray.length);

        for (int i = 0; i < newArray.length; i++) {
            System.out.print(newArray[i] + " ");
        }
        System.out.println();
    }
}
```

Enhanced For Loop

- ▶ 객체를 좀 더 쉽게 처리할 수 있는 향상된 for 문을 제공(Java 5 이상)
- ▶ 카운터 변수, 증감식을 사용하지 않고, 배열 및 컬렉션 항목의 개수만을 반복

```
public class EnhancedFor {  
    public static void main(String[] args) {  
        char vowels[] = { 'a', 'e', 'i', 'o', 'u' };  
        for (char vowel: vowels) {  
            System.out.print(vowel + " ");  
        }  
        System.out.println();  
    }  
}
```