

# JAVA 개발자 양성과정

(자바 프로그래밍 입문)

## 자바 소개

- 1995 년 Sun Microsystems 에서 발표
- 오라클(Oracle)이 2010 년에 Sun Microsystems 를 인수

## 자바 응용

### ● 웹 애플리케이션

- 자바는 웹 개발을 위한 Servlet, JSP 등을 제공한다.

### ● 엔터프라이즈 애플리케이션

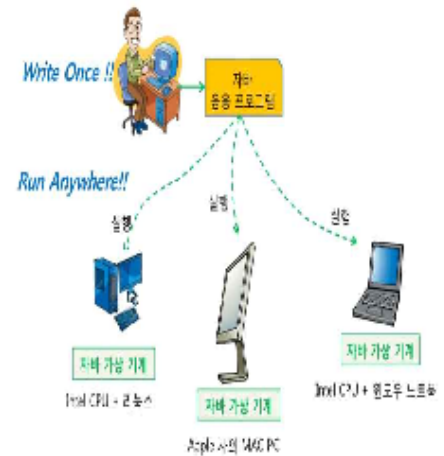
- 자바는 엔터프라이즈 애플리케이션을 개발하기 위한 Java EE(Java Enterprise Edition) 플랫폼을 제공한다.
- ERP(Enterprise Resource Planning) 시스템, 고객자원관리(CRM) 시스템 등

### ● 모바일 애플리케이션

- 자바는 모바일 애플리케이션을 빌드하기 위한 크로스 플랫폼 프레임워크인 J2ME 기능을 제공한다.
- 모바일 운영체제인 Android가 Java 기반 Android SDK를 사용하여 개발되었다.

## 자바의 특징

- 객체 지향 프로그래밍 언어(OOP) 이다.
- 플랫폼 독립적이다.
  - WORA(Write Once Run Anywhere)
- 보안에 강하다.
- 견고하다.
  - 자동 가비지 컬렉션 및 예외 처리
  - 컴파일과 런타임 시에 오류 검사



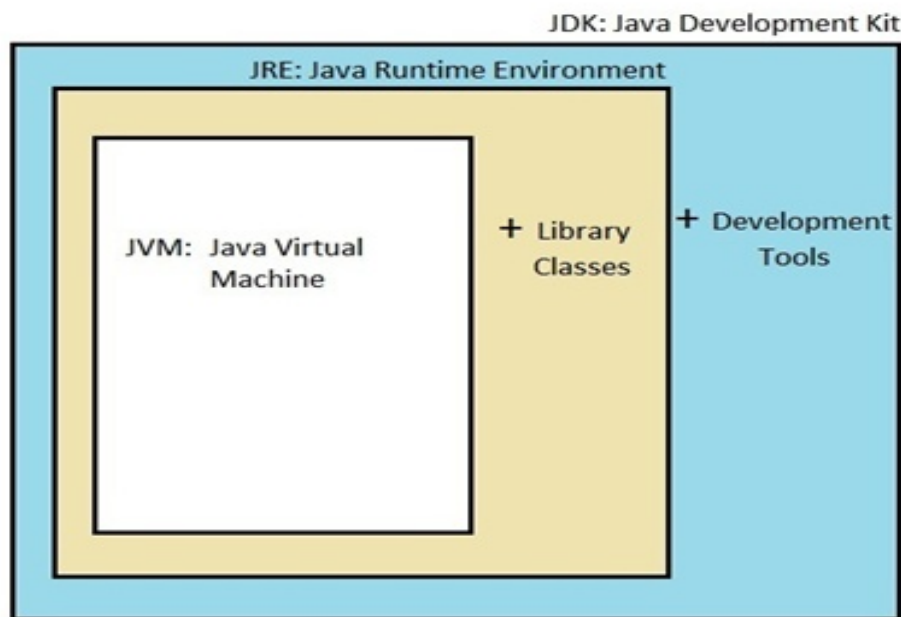
출처 : <https://usefultoknow.tistory.com/m/22>

### ※ 플랫폼이란?

- 운영체제와 CPU 아키텍처를 말한다.

## 자바 개발 도구(JDK) 설치

- 자바 애플리케이션을 개발하는데 필요한 도구와 환경을 제공한다.



*JDK = JRE + Development Tools*

*JRE = JVM + Library Classes*

출처 : [https://velog.io/@sung\\_hyuki/JVM-JRE-JDK](https://velog.io/@sung_hyuki/JVM-JRE-JDK)

## JDK 의 종류

### ● Open JDK : <http://openjdk.java.net/>

- 개발, 학습, 상업용 모두 무료(오픈소스기반)로 사용

### ● Oracle JDK : <https://www.oracle.com/java/>

- 개발, 학습용은 무료
- 상업용으로 사용할 경우 연간 사용료 지불
- 장기 기술 지원(LTS : Long Term Support)으로 안정적이다.

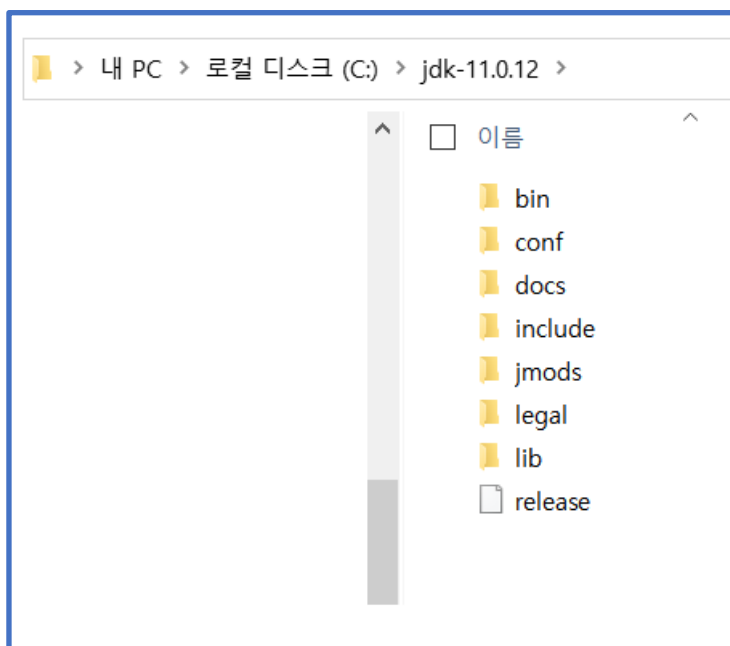
## Oracle JDK 설치 (jdk-11.0.12)

1. <https://developer.oracle.com/kr/> 페이지로 이동한다.
2. 상단 메뉴에서 'Downloads'를 선택한다.
3. 'All Java Downloads'를 선택한다.
4. 페이지 상단에서 Java 에서 'Java(JDK) for Developers' 를 선택한다.
5. Java SE 11 (LTS) 에서 Oracle JDK 에서 JDK Download 를 선택한다.

단, 오라클 계정이 없는 경우 계정을 생성한 후 라이선스에 동의한 후 다운로드한다.

- jdk-11.0.12\_windows-x64\_bin.exe 실행파일을 다운로드한다.

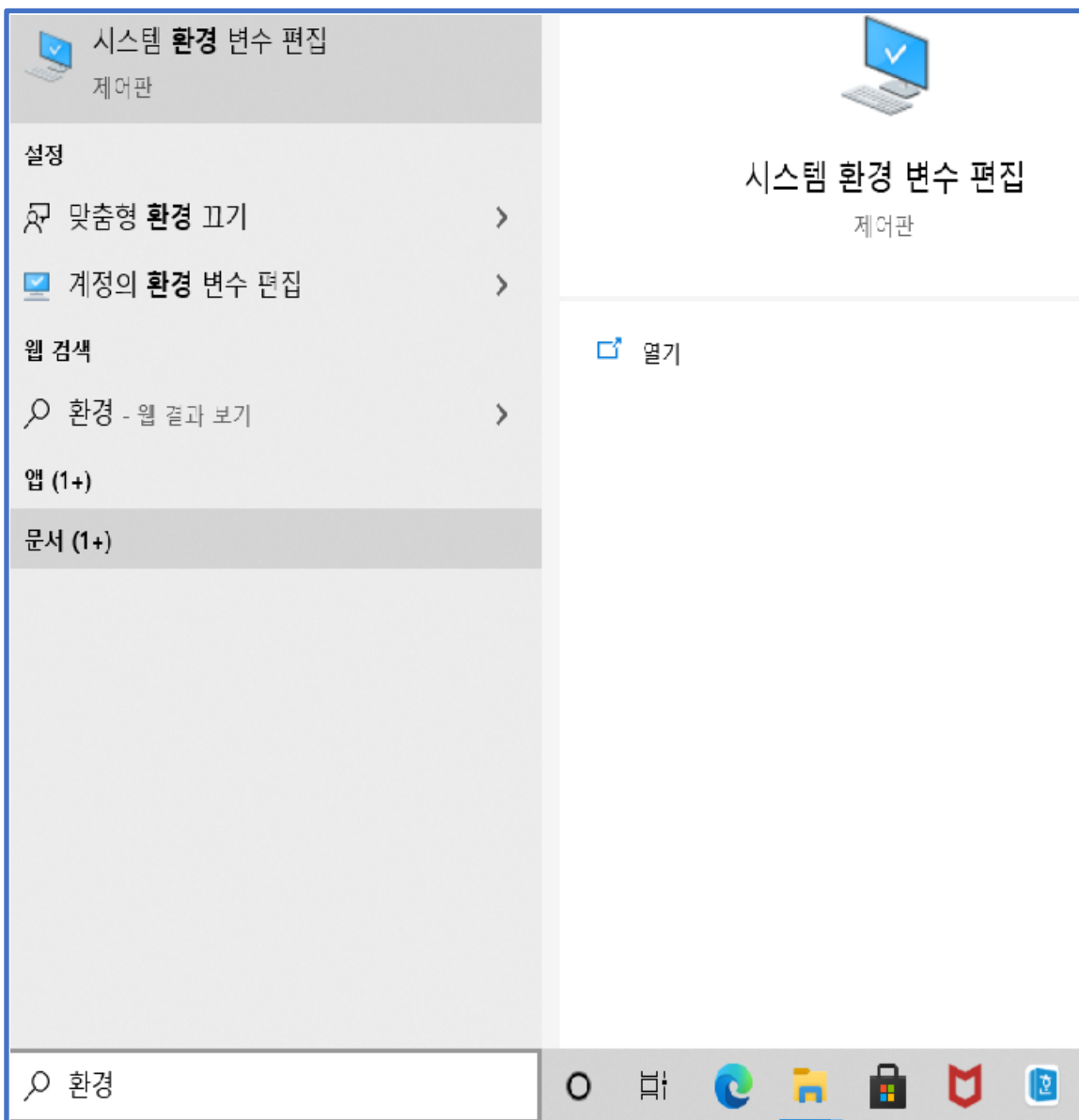
※ 설치 경로 : C:\jdk-11.0.12



## 환경변수 설정

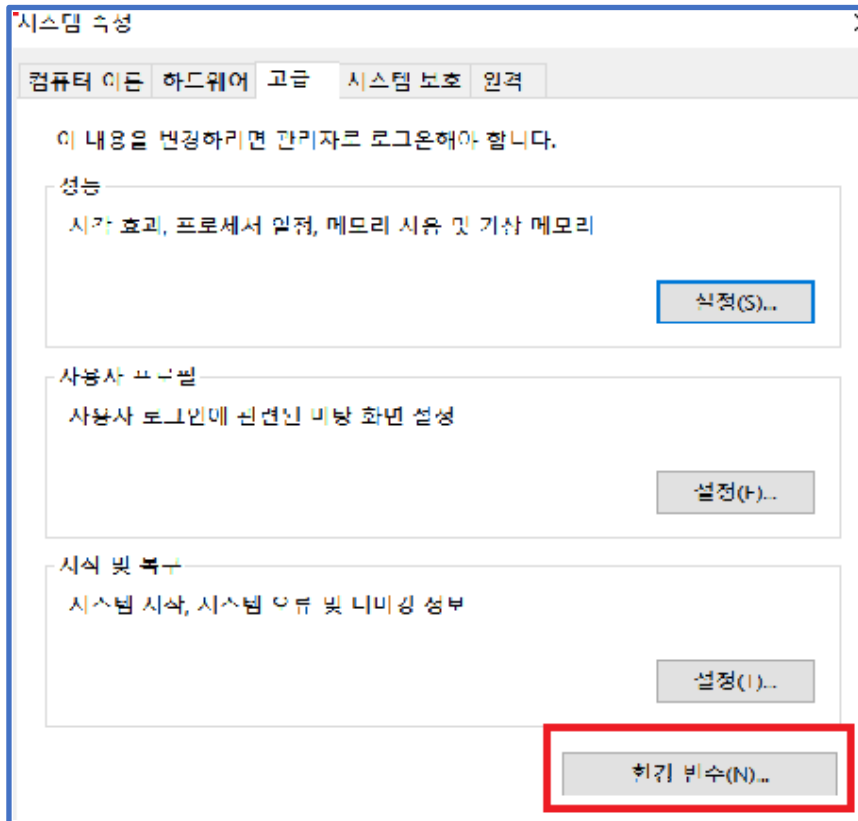
- OS(운영체제) 에서 사용하는 변수를 환경변수라고 한다.

1. **환경** 이라고 입력한 후 **시스템 환경 변수 편집**을 선택한다.

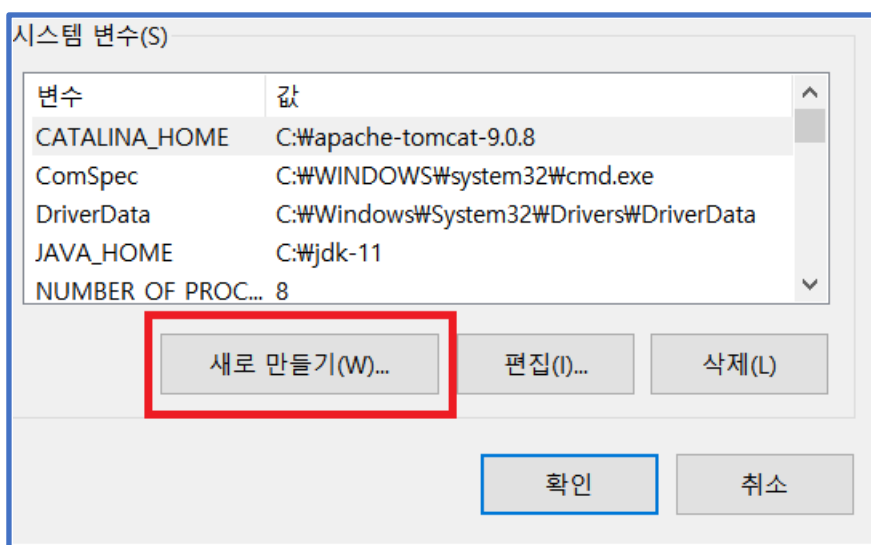




## 2. 환경 변수(N)... 버튼을 선택한다.

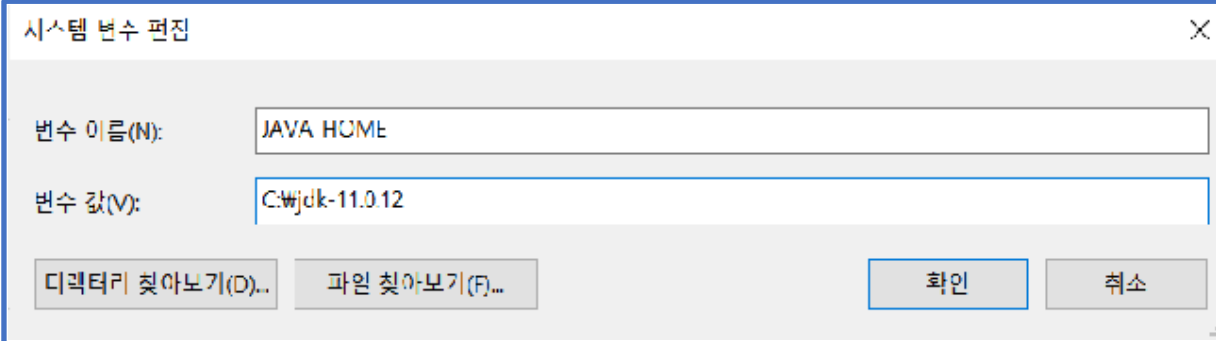


## 3. 시스템 변수 에서 새로 만들기(W)... 버튼을 선택한다.



#### 4. 변수 이름 : JAVA\_HOME

변수 값 : jdk 설치 경로



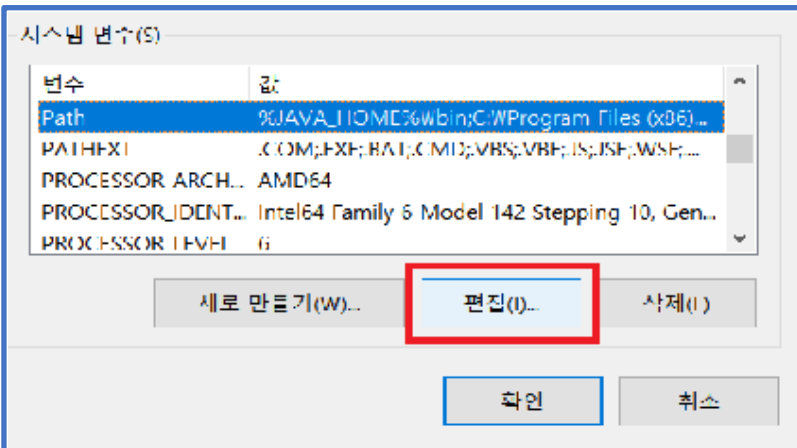
시스템 변수 편집

변수 이름(N): JAVA\_HOME

변수 값(V): C:\jdk-11.0.12

디렉터리 찾아보기(D)... 파일 찾아보기(F)... 확인 취소

#### 5. 시스템 변수에서 Path 를 선택하고 편집(I)... 버튼을 선택한다.



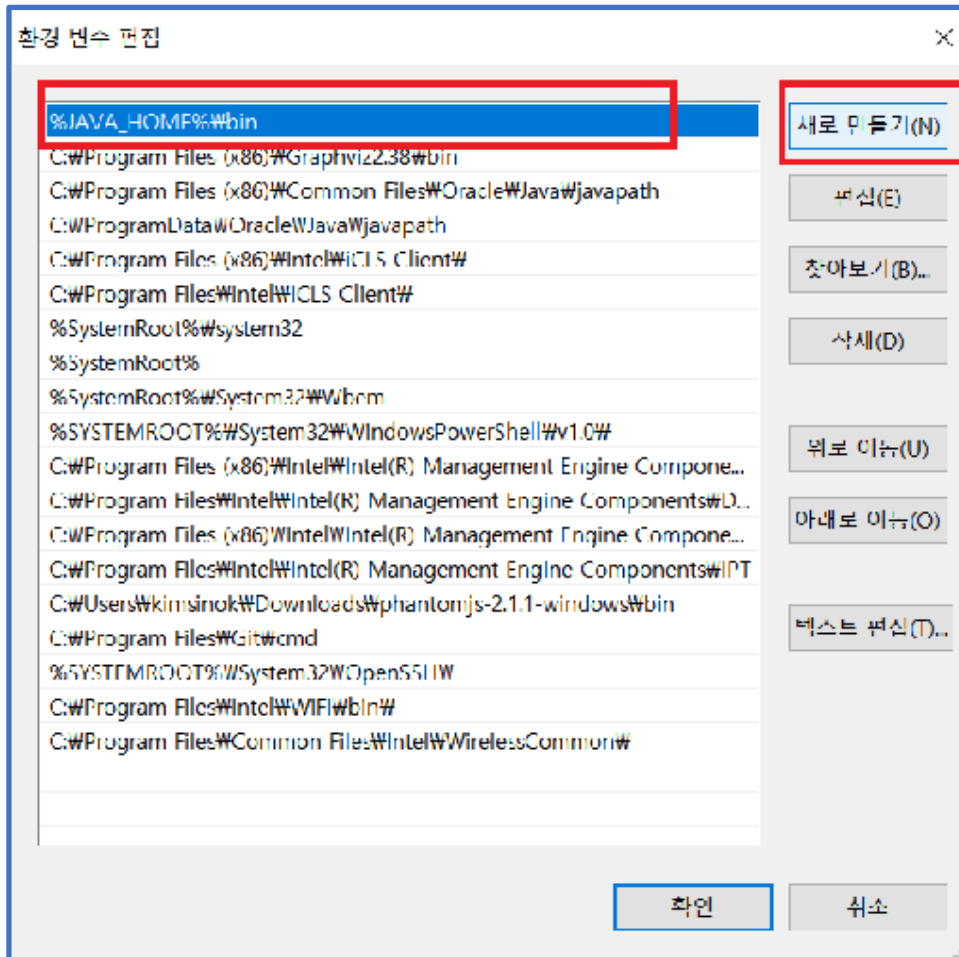
시스템 변수(S)

변수	값
Path	%JAVA_HOME%\bin;C:\Program Files (x86)...
PATH	COM;EXE;BAT;CMD;VBS;VB;JS;JSE;WSH;...
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 142 Stepping 10, Gen...
PROCESSOR_LEVEL	6

새로 만들기(W)... 편집(I)... 삭제(D)

확인 취소

6. 새로 만들기 버튼을 선택한 후 **%JAVA\_HOME%\bin** 을 입력한 후 맨 위로 이동시킨다.



7. 확인 버튼을 선택한다.

## JDK 버전 확인 - CLI (Command-line interface)

1. 시작 메뉴에서 명령 프롬프트를 선택한다. (검색창에 cmd 를 입력한다.)

2. 명령 프롬프트 창에서 **set** 을 입력한다.

- **set** : 명령 프롬프트 환경에서 환경변수 설정 및 확인하는 명령어

- 환경 변수 JAVA\_HOME, Path 를 확인한다.

- set path

- set JAVA\_HOME

3. JDK 버전 확인

**java -version**

# 간단한 자바 프로그램 작성하기

1. 소스파일을 작성한다. (Hello.java)

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

2. Java Compiler ( javac )를 사용하여 자바 소스 파일을 컴파일한다.

컴파일을 하면 자바 바이트 코드(Hello.class 파일)가 생성된다.

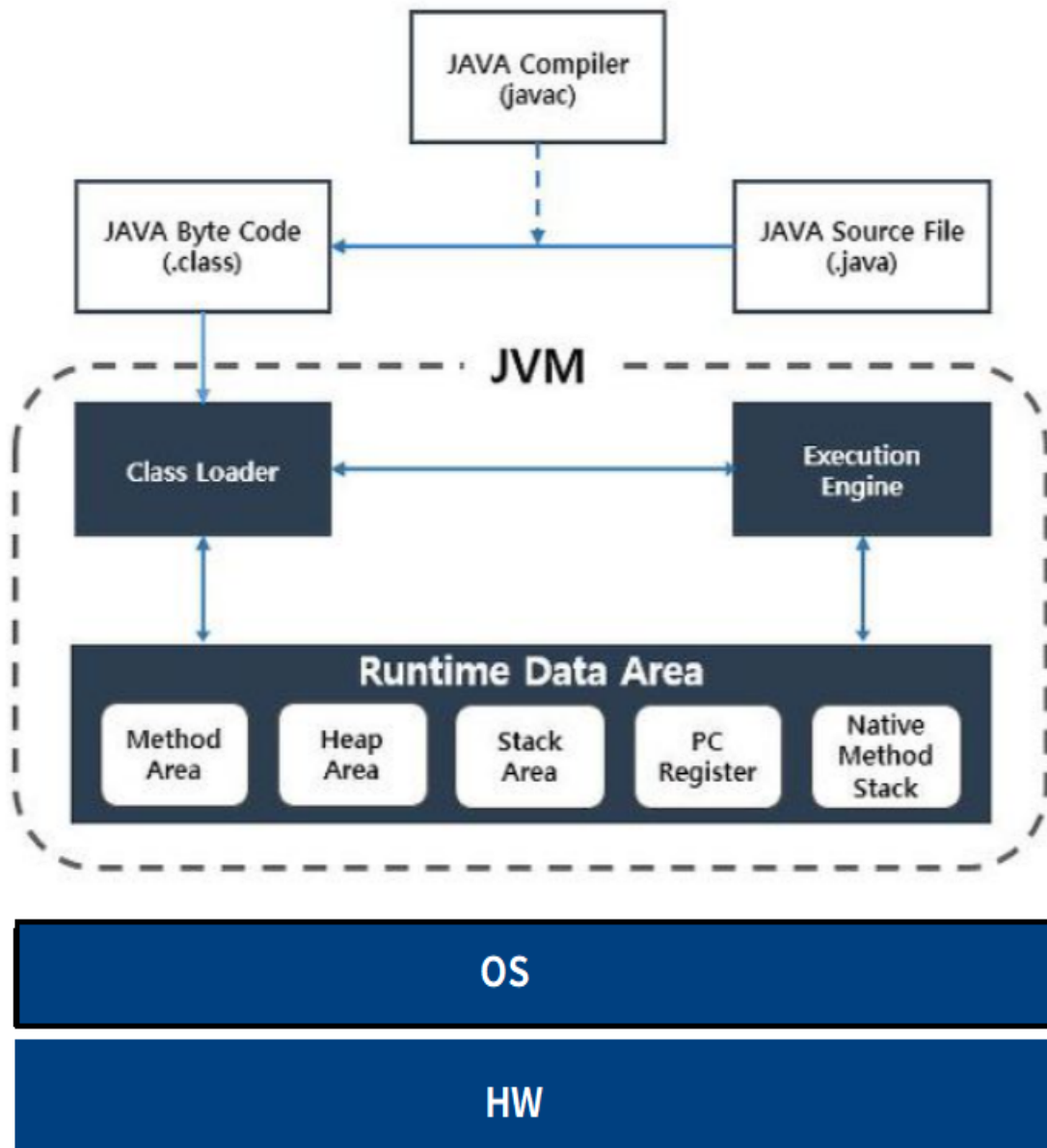
```
C:\Wkso> javac Hello.java
```

3. 자바 명령어( java command)는 자바 바이트 코드를 해석(interpret)하고 실행한다.

이 과정에서 바이트 코드가 바이너리 코드로 변경된다.

```
C:\Wkso> java Hello
```

## 자바 프로그램 실행과정 및 JVM 의 구조



## ● Java Compiler ( javac )

- Java 소스 파일은 JVM 이 해석할 수 있는 자바 바이트 코드(.class 파일)로 변경된다.

## ● Class Loader

- JVM 내로 .class 파일을 로드한다. 로딩된 클래스들은 Runtime Data Area 에 배치된다.

## ● Execution Engine

- 로딩된 클래스의 바이트 코드를 해석(interpret)하고 실행한다. 이 과정에서 바이트 코드가 바이너리 코드로 변경된다.

## ● Runtime Data Area

- JVM 이 OS(운영체제)로 부터 할당받은 메모리 영역이다.

# Runtime Data Area

- JVM 이 프로그램을 수행하기 위해서 OS(운영체제)로 부터 할당받은 메모리 영역이다.

## ● Method Area

- Class 와 Interface 의 자바 바이트 코드 및 메타 데이터가 저장된다.
- Field Information, Method Information, Class Variable(클래스 변수), 상수
- Java 8 이후로는 Metaspace 라는 OS 가 관리하는 영역으로 변경되었다.

## ● Heap Area

- new 명령어로 생성된 인스턴스가 저장되는 영역
- Garbage Collection 의 대상이 되는 영역

## ● Stack Area

- 메소드 내에 사용되는 값들(매개변수, 지역변수, 리턴값 등)이 저장되는 영역.

## ● PC Register

- CPU 의 register 와 역할이 비슷하다. 현재 수행중인 JVM 명령의 주소값이 저장된다.

## ● Native Method Stack

- 다른 언어(C/C++ 등)로 작성된 네이티브 코드들을 위한 영역
- JNI 를 통해 호출되는 C/C++ 코드



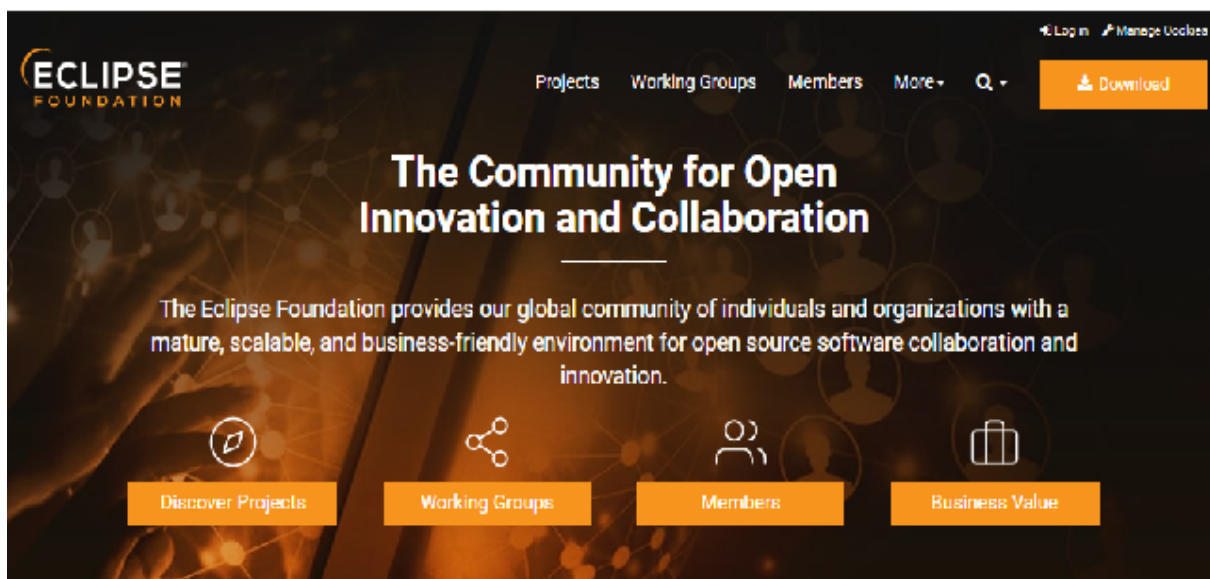
# Eclipse 개발 환경 구축

Eclipse 란?

- 무료 오픈 소스 통합 개발 환경(IDE : Integrated Development Environment) 도구이다.
- IDE : 프로젝트 생성, 자동 코드 완성, 디버깅 등과 같이 개발에 필요한 여러가지 기능을 통합적으로 제공해주는 도구이다.


Eclipse 설치 : Eclipse IDE 2020-03 R Packages

1. <https://www.eclipse.org> 사이트로 이동한다.
2. Download 버튼을 선택한다.



### 3. Download Packages 선택한다.

The Eclipse Installer 2021-06 R now includes a JRE for macOS, Windows and Linux.



## Get **Eclipse IDE 2021-06**

Install your favorite desktop IDE packages.

[Download x86\\_64](#)

[Download Packages](#) [Need Help?](#)

4. Eclipse IDE for Enterprise Java Developers 에서 운영체제에 맞는 항목을 선택한다.



Eclipse IDE 2020-03 R Packages

**Eclipse IDE for Java Developers**  
156 MB 592,857 DOWNLOADS  
The essential tools for any Java developer, including a Java IDE, a Git client, WSL Editor, Mylyn, Maven and Gradle integration.

**Eclipse IDE for Enterprise Java Developers (includes Incubating components)**  
400 MB 150,184 DOWNLOADS  
Tools for developers creating Java Enterprise and Web applications. Including a Java IDE, tools for Web Services, JPA and Data, tools, JST, Mylyn, Maven and Gradle, Git, and more.  
Click here to file a bug against Eclipse Web Tools Platform.  
Click here to file a bug against Eclipse Platform.  
Click here to file a bug against Maven integration for web projects.  
Click here to report an issue against Eclipse Wild Web Developer (incubating).

Windows x86\_64  
macOS x86\_64  
Linux x86\_64

Windows x86\_64  
macOS x86\_64  
Linux x86\_64

5. eclipse-jee-2020-03-R-incubation-win32-x86\_64.zip 파일을 다운로드한다.



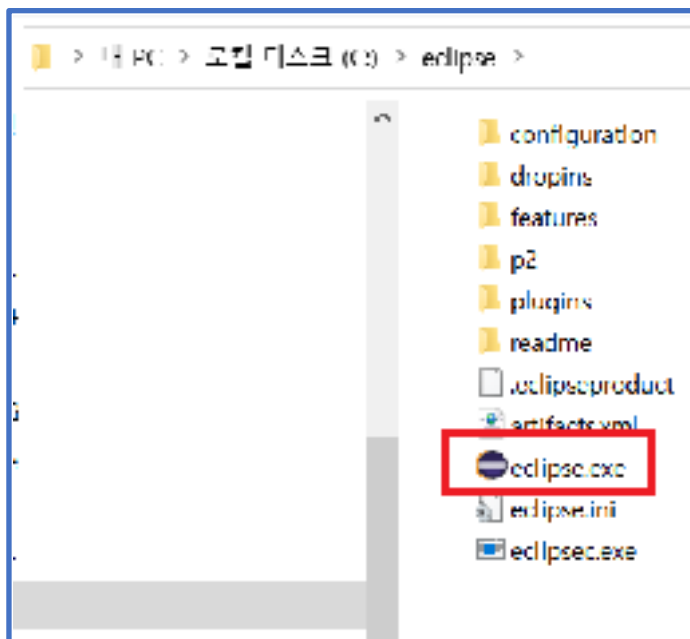
**Download**

Download from: Japan - Japan Advanced Institute of Science and Technology (https)

File: eclipse-jee-2020-03-R-incubation-win32-x86\_64.zip SHA-512

>> Select Another Mirror

6. C:\eclipse 폴더에 설치한다.

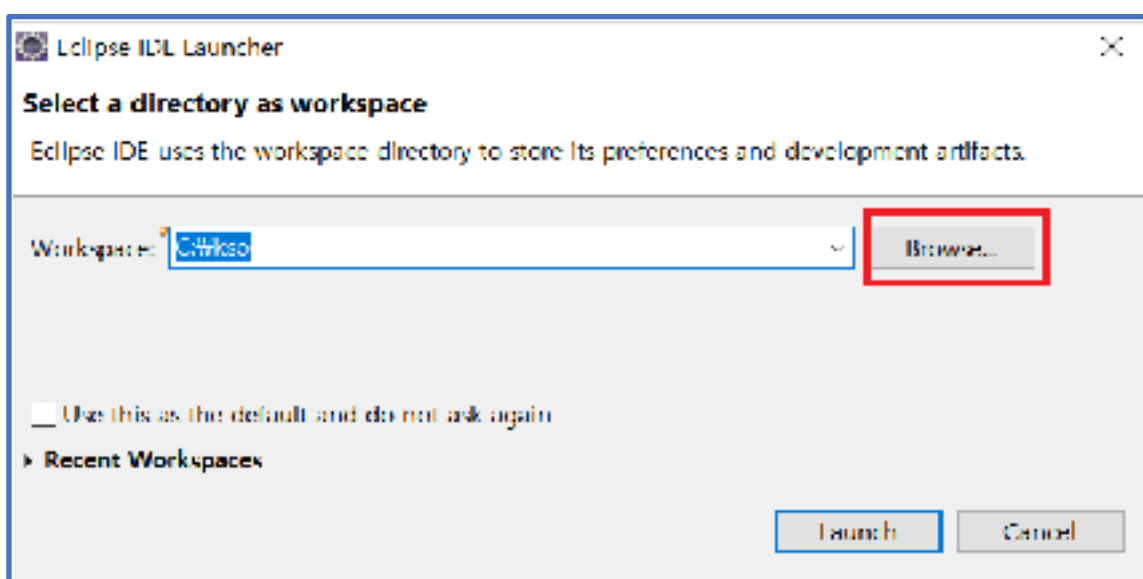


7. eclipse.exe 파일을 실행한다.

8. Browse... 버튼을 클릭하여 워크스페이스를 선택한 후 Launch 를 선택한다.

- 워크스페이스 : 프로젝트를 저장하는 작업 폴더이다.

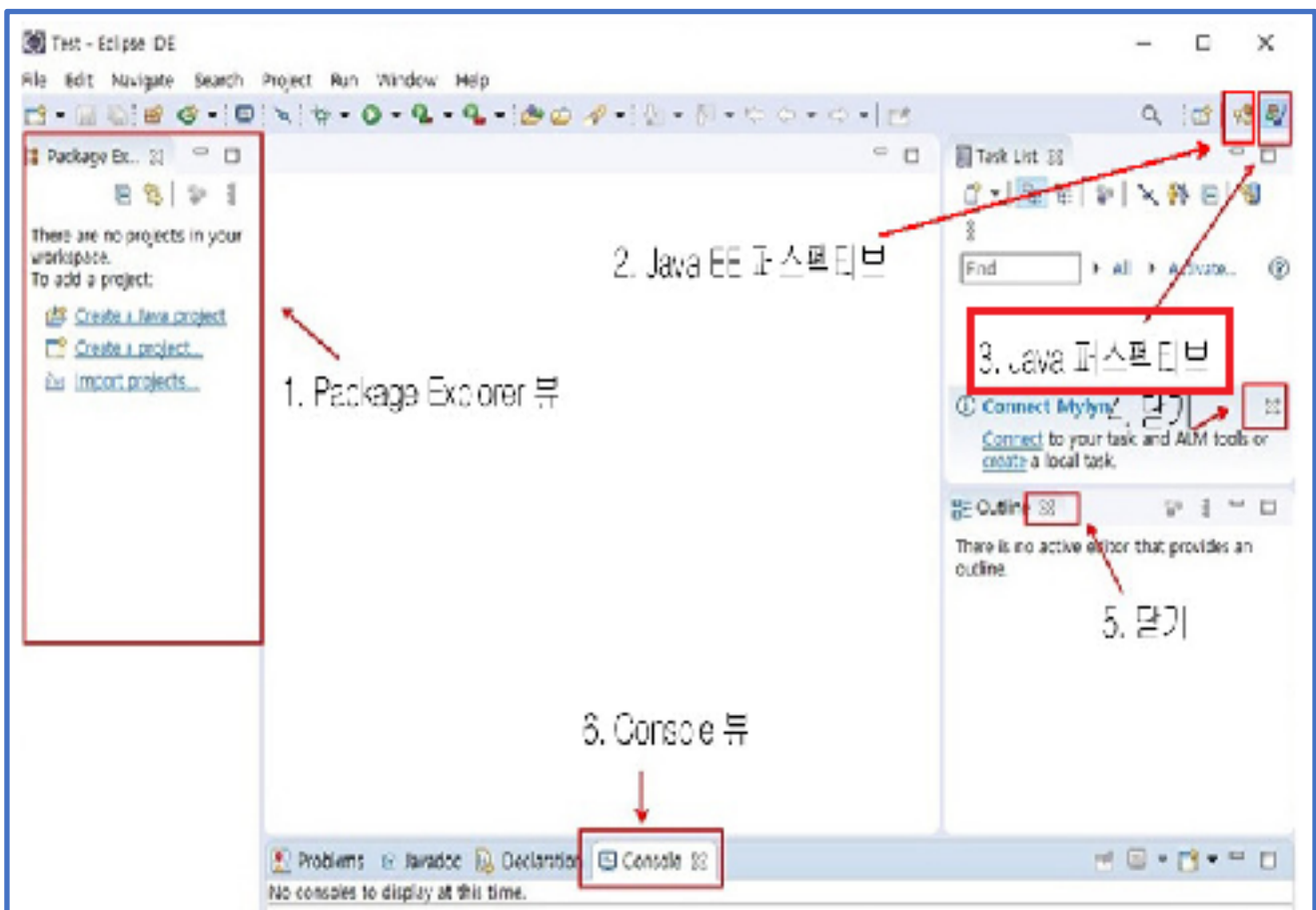
C:\본인이름폴더(ex : kso)



## Eclipse 화면 구성

### • 퍼스펙티브와 뷰

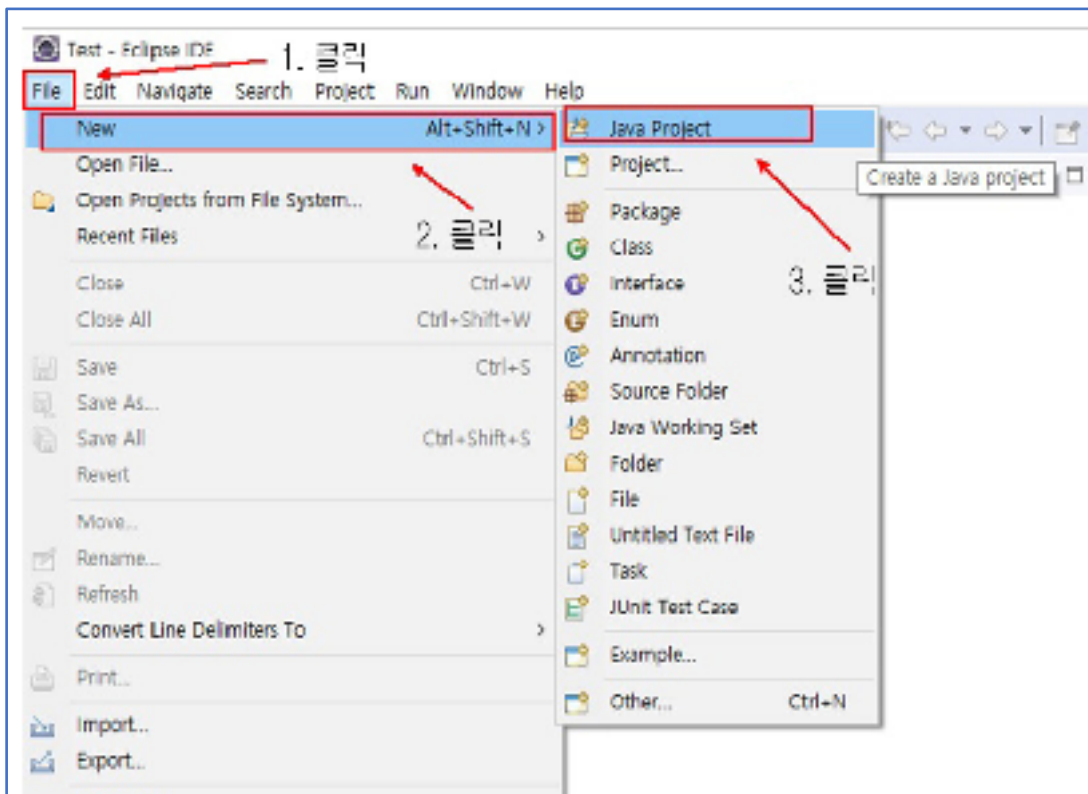
퍼스펙티브는 이클립스에서 유용하게 사용할 수 있는 뷰들의 묶음을 말하며, 뷰는 이클립스 내부에 있는 작은창을 말한다.



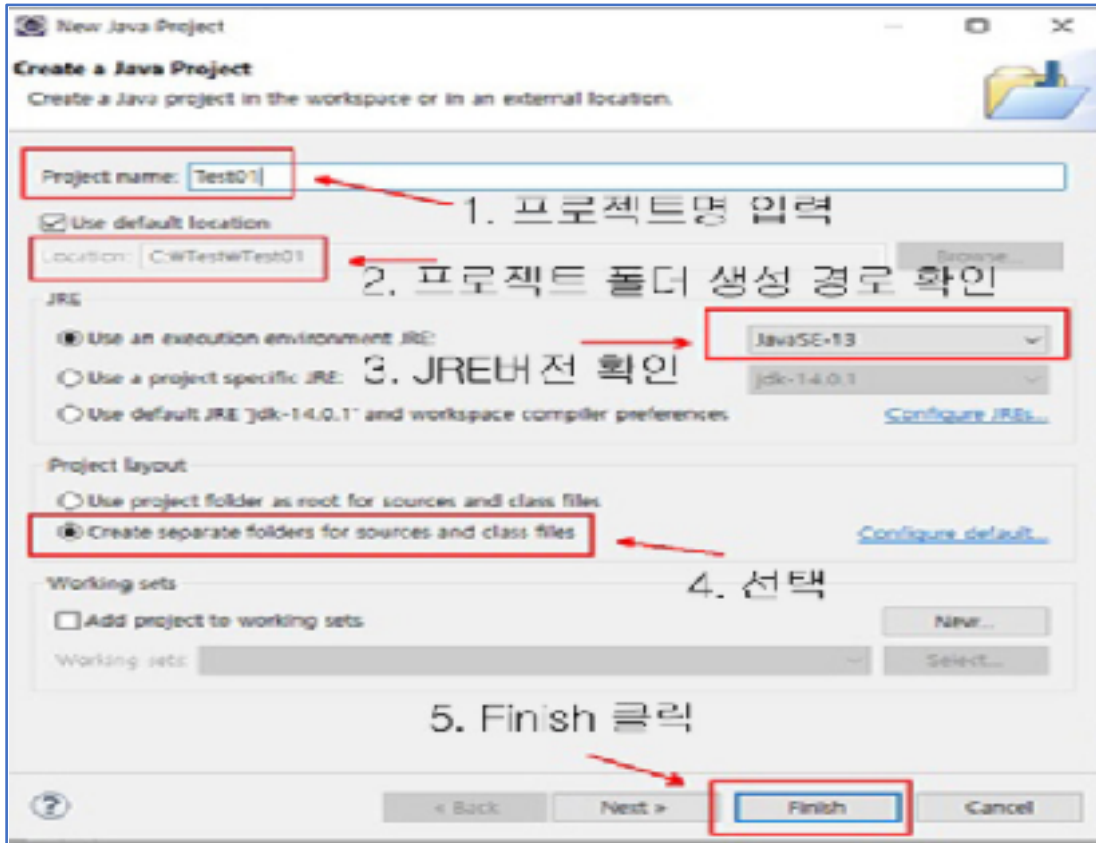
- Java 퍼스펙티브
- Package Explorer 뷰
- Console 뷰

# Eclipse 에서 Java Project 생성 및 실행

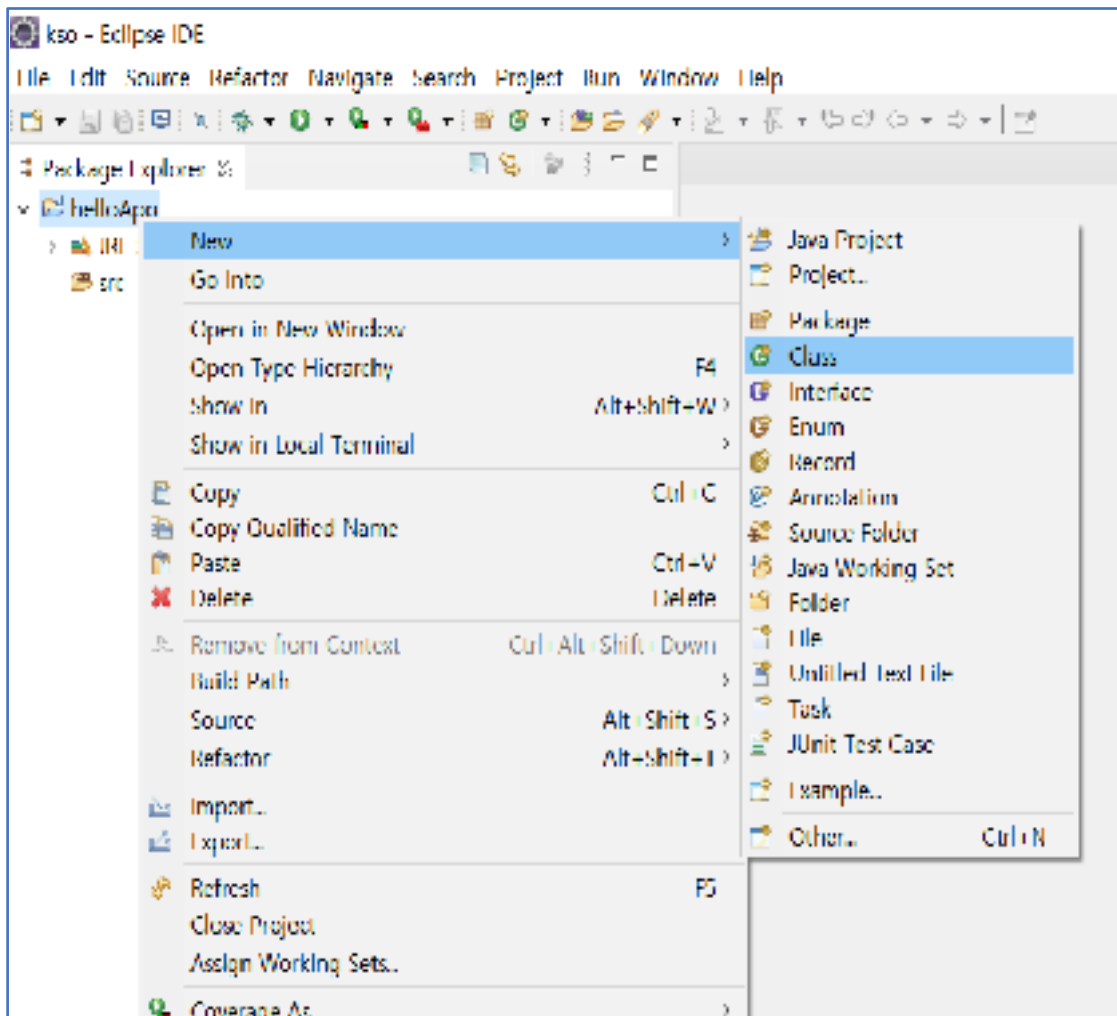
## 1. 자바 프로젝트 생성



- Project name : HelloApp



## 2. 소스파일 생성 : Hello.java



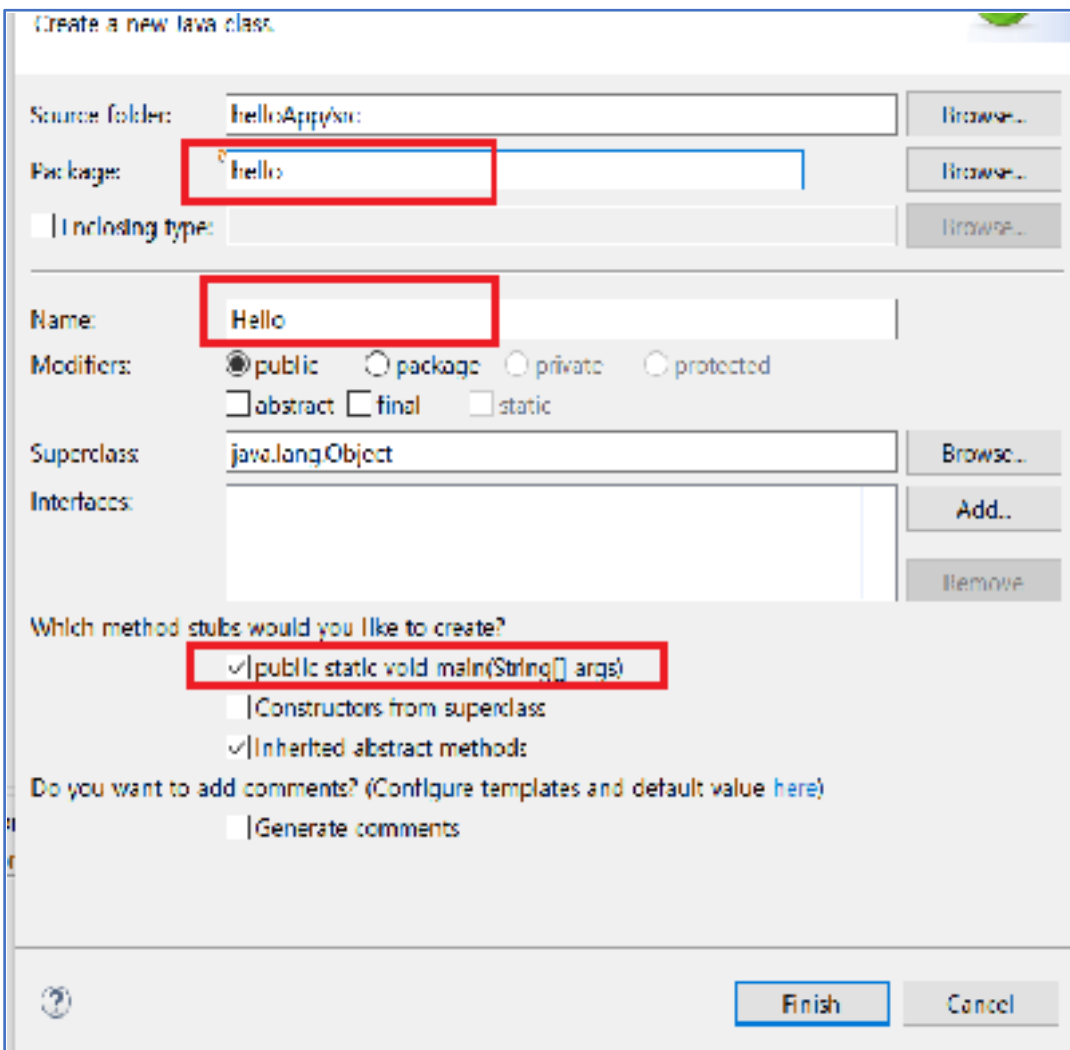


## ● 자바 클래스 생성

- Package : 파일들을 기능별로 분류하기 위한 폴더를 말한다.

패키지명 (명명규칙 : 첫글자는 소문자로 시작한다.) hello

- Name : 클래스명 (명명규칙 : 첫글자는 대문자로 시작한다.) Hello
- public static void main(String[] args) 항목을 체크한다.



Create a new Java class

Source folder:  

Package:  

Including type:  


---

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static

Superclass:  

Interfaces:

Which method stubs would you like to create?

☒ public static void main(String[] args)

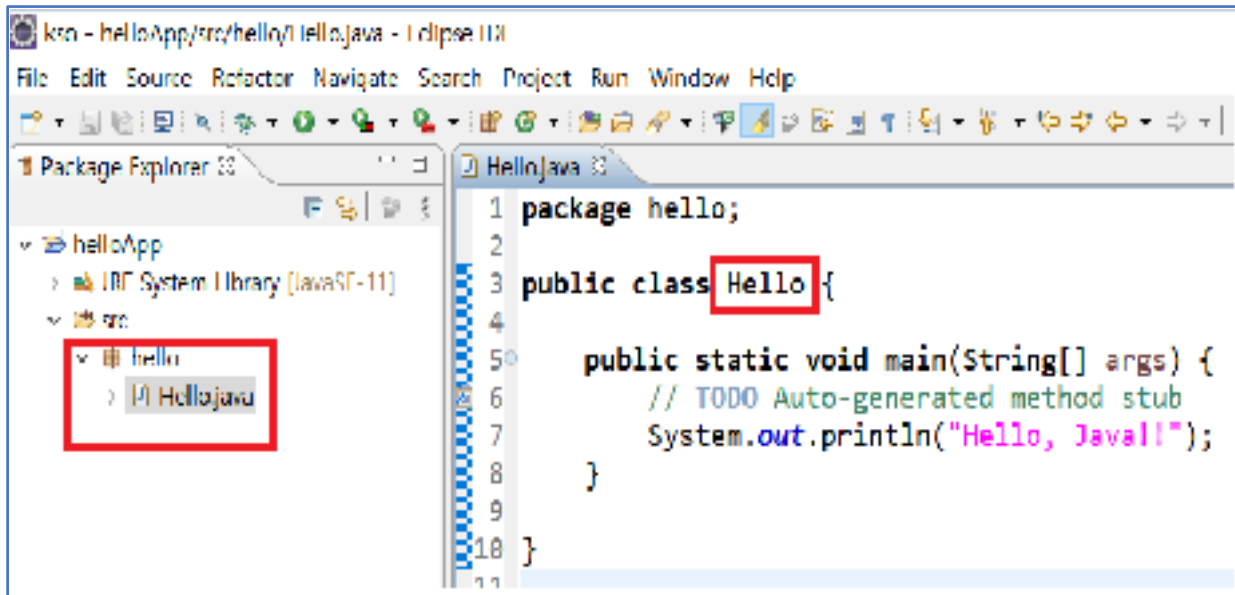
☐ Constructors from superclass

☒ Inherited abstract methods

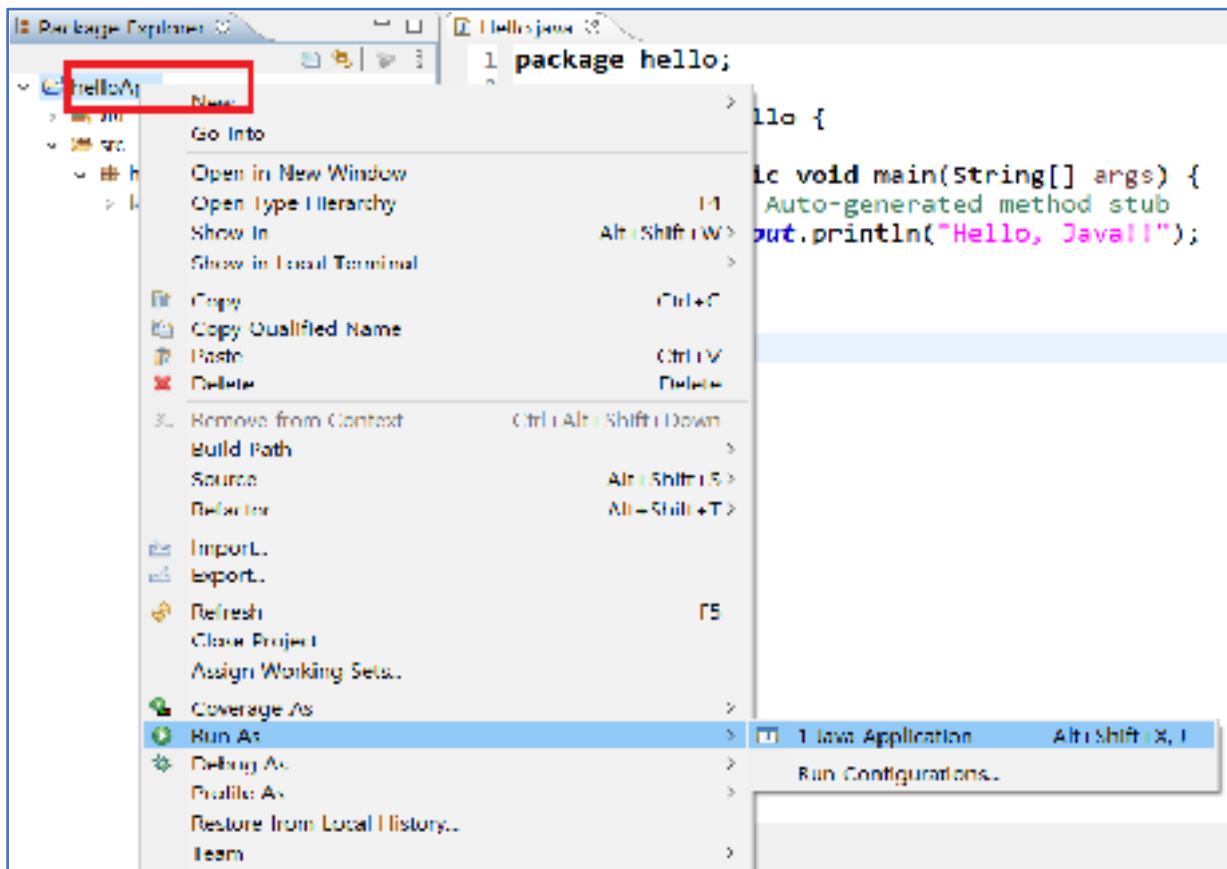
Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

## ● 소스 코드 작성



## ● 프로그램 실행 : CTRL + F11 (단축키)



## 모듈(Module)

### 모듈(Module)이란?

- 자바 9 버전 부터 지원하는 기능이다.
- 모듈은 패키지보다 상위의 개념이며, 모듈별로 패키지를 나누어 관리함으로써 캡슐화를 달성하고, 필요에 따라 조합하는 기법이다.

### 모듈의 이점

1. 군더더기를 제거함으로써 애플리케이션이 알맞은 덩치를 갖게한다.
2. 패키지의 캡슐화를 가능하게 한다.
3. 필요한 모듈이 있는지 없는지를 JVM start-up 타임에 검사한다.

## 모듈 기술자(Module Descriptor)의 구조 : module-info.java

```
module 모듈명 {  
    requires 모듈명;  
    exports 패키지명;  
}
```

### 1. 모듈명

모듈명은 unique 하고, 가능하면 모듈속 자바 패키지의 이름과 같게 하는 것이 좋다.

### 2. exports

패키지 단위로 다른 모듈에서 사용할 수 있도록 공개하는 키워드이다.

### 3. requires

의존하고 있는 모듈을 지정하는 키워드이다.

기본적으로 모든 모듈은 java.base 라는 모듈을 의존한다.

## 주석 (Comment)

1. 프로그램 소스 코드에 부연 설명을 달 때 사용한다.
2. 주석은 프로그램 수행에 영향을 미치지 않는다. 왜냐하면 컴파일 시 제외된다.
3. 자바에는 두 가지 형태의 주석이 있다.

- 라인 주석 : 이클립스 주석달기 단축키 ( Ctrl + / )

```
int age;    // 나이
```

- 블록 주석 : 이클립스 주석달기 단축키 ( Ctrl + Shift + / )

```
/*
```

```
    작성자 : 홍길동
```

```
    작성일자 : 2021-01-01
```

```
    프로그램 설명 :
```

```
*/
```

```
public class MyProgram {
```

```
    ...
```

```
}
```

## 변수 (Variable)

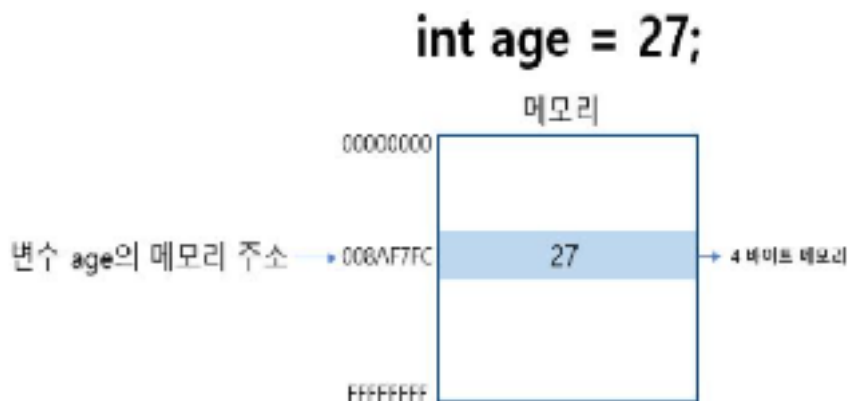
- 프로그램을 실행하는데 사용되는 데이터는 메모리(RAM)에 저장된다.
- 변수는 데이터를 저장할 수 있는 메모리 공간을 의미하며, 저장된 값은 변경될 수 있다.

변수 선언

**int age;**

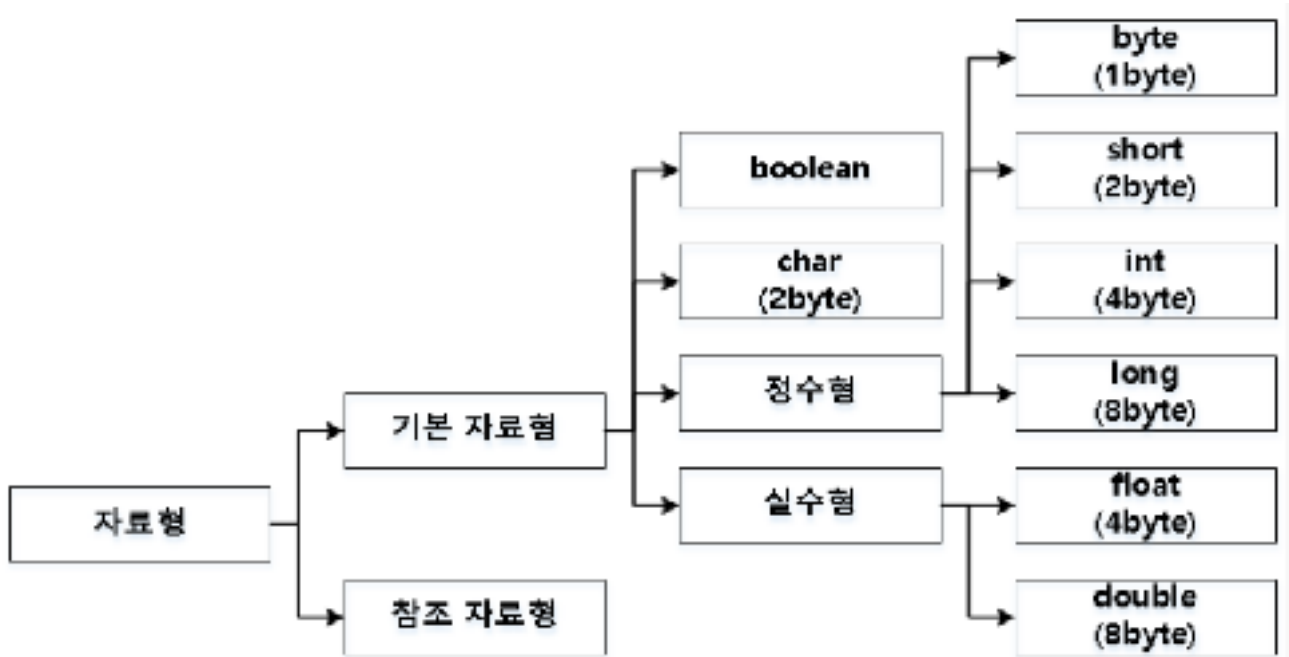
자료형    변수이름

변수와 메모리



- 하나의 메모리 공간에는 1 바이트의 데이터가 저장된다.
- 자료형이 int 이므로 4 바이트 공간이 할당된다.
- 변수의 이름은 첫 번째 메모리 주소인 008AF7FC 만을 가리키게 된다.

## 자료형 (Data Type)



## 기본 자료형 (Primitive Data Type)

- 자바에서 여러 형태의 데이터 타입을 미리 정의(Built-in)하여 제공한다.
- 실제 값(리터럴)을 저장하는 자료형이다.

### ● char 형

- 문자 리터럴은 유니코드로 변환되어 저장된다.
- char 는 정수 타입이므로 10 진수, 16 진수 형태의 유니코드 저장 가능하다.

### ※ 문자 인코딩 (character encoding)

- 문자 셋(문자 집합, Character set, Charset)은 정보를 표현하기 위한 글자나 기호들의 집합을 정의한 것이다.
- 문자 인코딩은 문자 셋을 컴퓨터에서 표현하기 위해 약속된 규칙에 따라 변환하는 과정을 말한다
- ASCII(아스키), 유니코드



## ● ASCII

· 미국 표준 정보 교환 코드

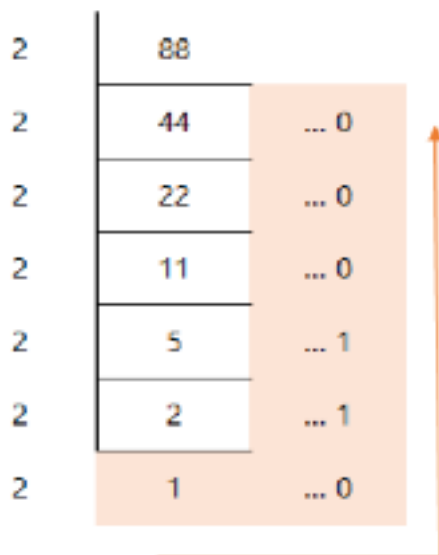
· 7 비트 표현 방식으로  $2^7$  총 128 개의 문자를 표현할 수 있다. ( 0- 127 코드값)

10진수	16진수	문자	10진수	16진수	문자	10진수	16진수	문자	10진수	16진수	문자
0	0x00	NUL	16	0x10	DXF	32	0x20	SP	48	0x30	0
1	0x01	SOH	17	0x11	DC1	33	0x21	!	49	0x31	1
2	0x02	STX	18	0x12	SC2	34	0x22	"	50	0x32	2
3	0x03	ETX	19	0x13	SC3	35	0x23	#	51	0x33	3
4	0x04	EOT	20	0x14	SC4	36	0x24	\$	52	0x34	4
5	0x05	ENQ	21	0x15	NAK	37	0x25	%	53	0x35	5
6	0x06	ACK	22	0x16	SYN	38	0x26	&	54	0x36	6
7	0x07	BEL	23	0x17	ETB	39	0x27	'	55	0x37	7
8	0x08	BS	24	0x18	CAN	40	0x28	(	56	0x38	8
9	0x09	HT	25	0x19	EM	41	0x29	)	57	0x39	9
10	0x0A	LF	26	0x1A	SUB	42	0x2A	*	58	0x3A	:
11	0x0B	VT	27	0x1B	ESC	43	0x2B	+	59	0x3B	;
12	0x0C	FF	28	0x1C	FS	44	0x2C	,	60	0x3C	<
13	0x0D	CR	29	0x1D	GS	45	0x2D	-	61	0x3D	=
14	0x0E	SO	30	0x1E	RS	46	0x2E	.	62	0x3E	>
15	0x0F	SI	31	0x1F	US	47	0x2F	/	63	0x3F	?

10진수	16진수	문자	10진수	16진수	문자	10진수	16진수	문자	10진수	16진수	문자
64	0x40	@	80	0x50	P	96	0x60	`	112	0x70	p
65	0x41	A	81	0x51	Q	97	0x61	a	113	0x71	q
66	0x42	B	82	0x52	R	98	0x62	b	114	0x72	r
67	0x43	C	83	0x53	S	99	0x63	c	115	0x73	s
68	0x44	D	84	0x54	T	100	0x64	d	116	0x74	t
69	0x45	E	85	0x55	U	101	0x65	e	117	0x75	u
70	0x46	F	86	0x56	V	102	0x66	f	118	0x76	v
71	0x47	G	87	0x57	W	103	0x67	g	119	0x77	w
72	0x48	H	88	0x58	X	104	0x68	h	120	0x78	x
73	0x49	I	89	0x59	Y	105	0x69	i	121	0x79	y
74	0x4A	J	90	0x5A	Z	106	0x6A	j	122	0x7A	z
75	0x4B	K	91	0x5B	[	107	0x6B	k	123	0x7B	{
76	0x4C	L	92	0x5C	\	108	0x6C	l	124	0x7C	
77	0x4D	M	93	0x5D	]	109	0x6D	m	125	0x7D	}
78	0x4E	N	94	0x5E	^	110	0x6E	n	126	0x7E	~
79	0x4F	O	95	0x5F	_	111	0x6F	o	127	0x7F	DEL

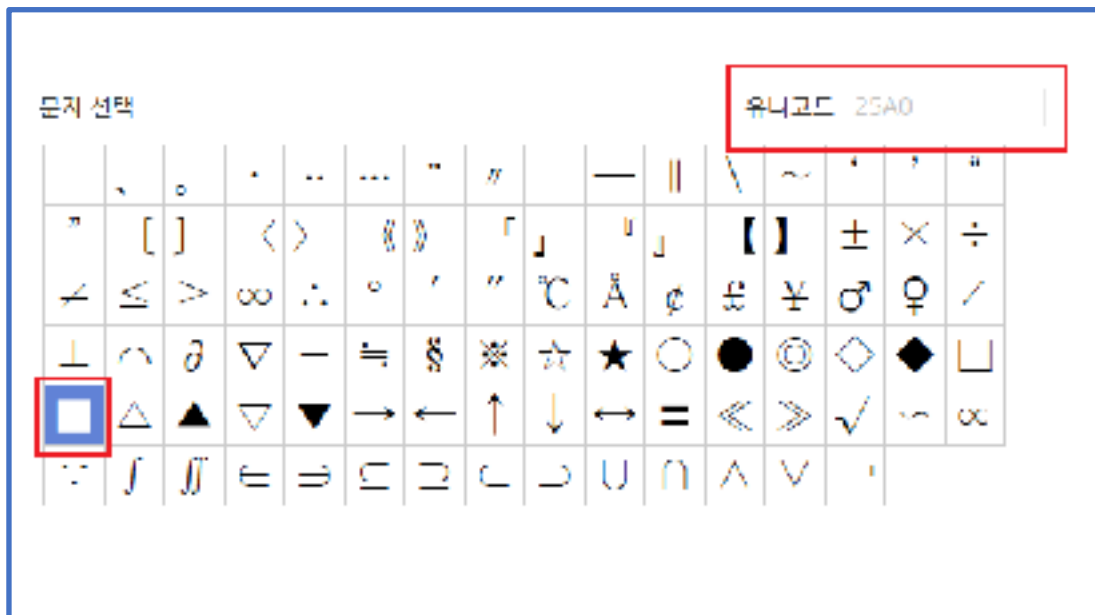
영문 대문자 'X' ----->>> 10 진수 아스키 코드값 (88)

10 진수 (88)----->>> 2 진수 변환 (1011000)



## ● UNICODE

- 전 세계의 모든 문자를 컴퓨터에서 일관되게 표현할 수 있다.
- 16 비트 표현 방식으로  $2^{16}$  총 65536 개의 문자를 표현할 수 있다. ( 0- 65535 코드값)



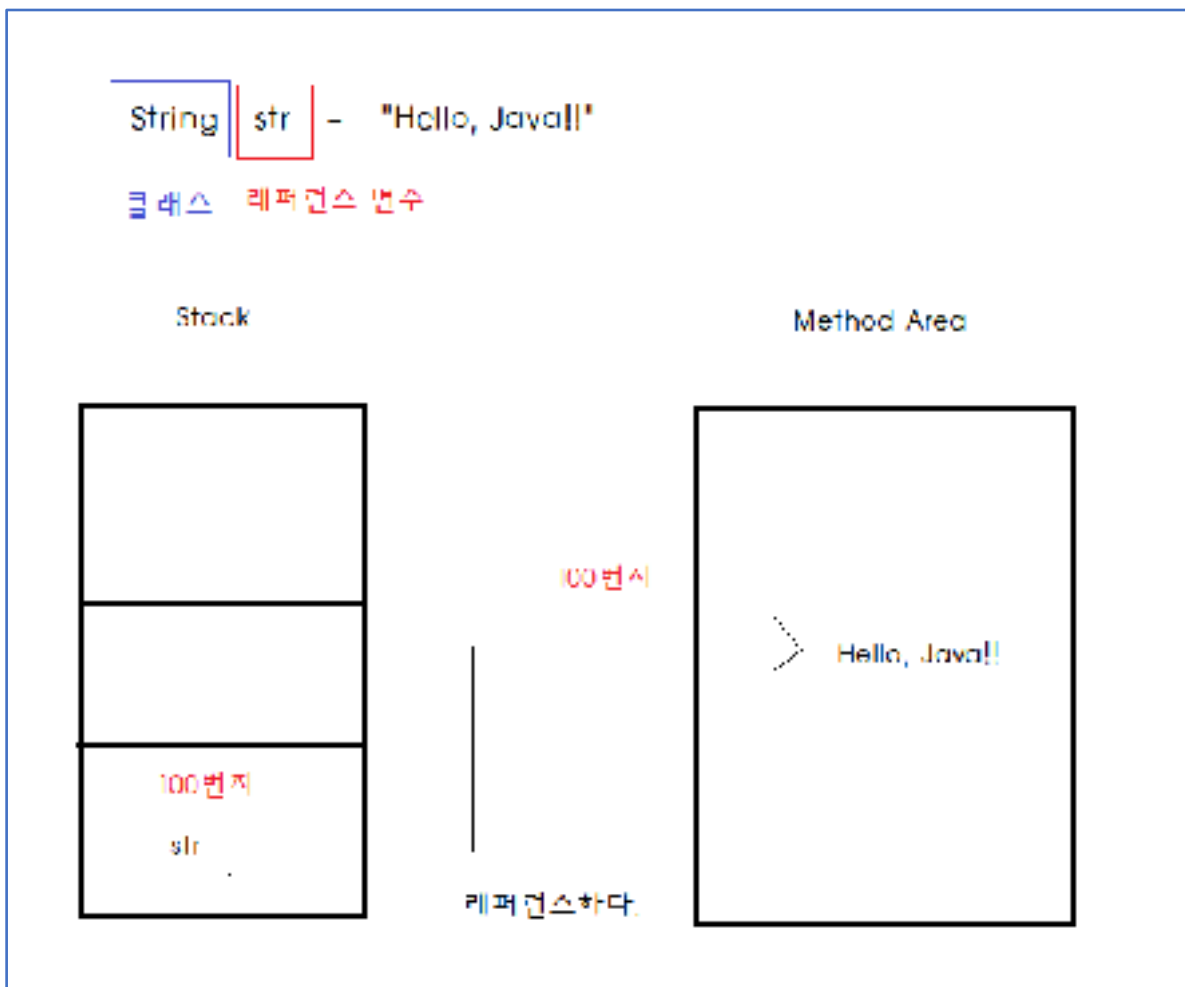
## Escape 문자

- 문자열 내에서 사용하는것으로 특수 기능을 제어할 때 사용한다.

$\backslash n$	개행 ( 줄바꿈 )
$\backslash t$	탭 ( 8 공백 )
$\backslash r$	작은 바운 ( 싱글쿼테이션 ) 표시
$\backslash "$	큰 바운표 ( 더블쿼테이션 ) 표시
$\backslash \backslash$	역슬래쉬 표시

## 참조형 (Reference Data Type)

- 객체의 저장 공간인 주소를 저장하고, 주소를 통해 객체를 참조하는 타입이다.
- 클래스, 인터페이스, 배열, 열거형(Enum)



## 리터럴 (literal)

- 소스 코드의 고정된 값이다.
- 정수, 실수, 문자, 논리, 문자열 리터럴 등이 있다.
- 정수형 리터럴

10 진수,

8 진수(숫자 0 으로 시작) 012

16 진수(숫자 0x) 0x123

2 진수(0b) 이 있다. 0b1

- 정수형 리터럴은 int 형으로 컴파일되고, long 형 리터럴은 숫자뒤에 L / l 을 붙인다.
- 실수형 리터럴은 double 형으로 컴파일되고, float 형 리터럴은 숫자뒤에 F / f 을 붙인다.
- 문자형 리터럴은 단일 인용부호를 붙인다.
- 문자열형 리터럴은 이중 인용부호를 붙인다.
- 논리형 리터럴은 true, false 가 있다.

## 타입 변환 (Data Type Conversion)

- 하나의 자료형을 다른 자료형으로 바꾸는 것을 말한다.
- 자바에서는 boolean 형을 제외한 나머지 기본 자료형간의 타입 변환을 수행할 수 있다.

### ● 자동 타입 변환(묵시적) : Promotion

- 작은 메모리 크기의 데이터 타입을 큰 메모리 크기의 데이터 타입으로 변환하는 것이다.
- 연산을 수행 시 컴파일러가 자동으로 타입 변환을 수행해준다.

---

byte (1) < short (2) < int (4) < long (8) < float (4) < double (8)

---

### ● 강제 타입 변환(명시적) : Casting

- 사용자가 캐스트 연산자를 사용하여 강제적으로 타입 변환을 수행한다.
- 작은 메모리 크기의 데이터 타입 = (작은 메모리 크기의 데이터 타입) 큰 메모리 크기의 데이터 타입
- 데이터의 손실이 발생할 수 있다.

## ● 자바 정수 연산에서의 자동 타입 변환

- 정수형 변수가 산술 연산식에서 피연산자로 사용되면 int 타입보다 작은 byte, short 형 변수는 int 형 으로 자동 타입 변환된다.
- 두 피연산자 중 int 형 보다 큰 경우 큰 자료형으로 변환되어 연산된다.

## ● + 연산에서의 문자열 자동 타입 변환

- 피연산자 중 하나가 문자열일 경우 나머지 피연산자도 문자열로 자동 변환되고 문자열 결합 연산을 수행한다.

## ● 문자열을 기본 자료형으로 강제 변환

String -> int	<pre>String str = "12345"; int num = Integer.parseInt(str);</pre>
String -> double	<pre>String str = "123.45"; double num = Double.parseDouble(str);</pre>

## 콘솔 입출력

### ● Scanner 클래스

- 읽은 바이트를 문자, 정수, 실수, 불린, 문자열 등 다양한 타입으로 변환하여 리턴한다.
- 입력되는 데이터를 공백(whitespace) 또는 개행(줄바꿈) 으로 구분되는 토큰 단위로 읽는다.

Scanner scan = new Scanner( System.in )

메소드명	설명
next( )	다음 토큰을 문자열로 반환 (공백을 기준으로 한 단어를 읽음 )
nextLine( )	문자열 반환 (개행을 기준으로 한 줄을 읽음 )
nextInt( )	다음 토큰을 int형 으로 반환
nextDouble( )	다음 토큰을 double형 으로 반환
hasNext( )	현재 입력된 토큰이 있으면 true, 아니면 무한 대기 상태 그러다 입력이 들어오면 true, ctrl z가 입력되면 false를 반환



## ※ Console 이란?

- 시스템을 사용하기 위해 키보드로 입력을 받고 화면으로 출력하는 소프트웨어로  
리눅스 or 유닉스 : 터미널 / 윈도우 : 명령 프롬프트 / 이클립스 : Console 뷰

## ※ 엔터(Enter)의 아스키코드값

- 엔터는 Windows 에서는 `\r\n` 이며 `\r` 은 아스키코드값 13 CR 을 나타낸다. CR 은 Carriage Return 의 약자로 커서 표시 위치를 같은 줄(행) 맨 앞의 위치로 복귀시키는 것을 의미한다.  
`\n` 은 아스키코드값 10 LF 를 나타내고, LF 는 Line Feed 의 약자로 모니터의 커서 위치나 프린터의 인쇄 위치를 한 줄 아래로 내리는 일을 의미한다.

## ● System.out.printf ("출력 서식", 출력할 내용)

### ※ 출력 서식 (format)

`%[argument_index$][flags][width][.precision]conversion`

- 출력 서식의 지시자(conversion)를 제외한 나머지는 생략 가능하다.
- width : 출력할 전체 자리수 지정(오른쪽 정렬). 예) `%3d`, 전체자리수가 3 인 정수
- (flags) 0 : 전체 자리수가 지정된 경우 왼쪽의 남은 자리에 0 을 출력. 예) `%03d`
- (flags) - : 전체 자리수가 지정된 경우 왼쪽 정렬하고 빈칸에 공백 출력.
- .precision : 소수점 아래 자리수 지정. 잘리는 소수점 자리수는 반올림하여 출력한다.

## ※ conversion

지시자	설명
%b	불리언(boolean) 형식으로 출력
%d	10진(decimal) 정수형 형식으로 출력
%o	8진(octal) 정수의 형식으로 출력
%x, %X	16진(hexa decimal) 정수의 형식으로 출력
%f	부동 소수점의 형식으로 출력
%e, %E	지수 표현식의 형식으로 출력
%c	문자로 출력
%s	문자열로 출력

## 연산자 (Operator)

### ● 단항 연산자

- 항이 하나인 연산자이다.

연산자	의미	사용법	설명
<code>+, -</code>	부호 연산자	<code>+var, -var</code>	변수 <code>var</code> 의 부호를 바꾼다.
<code>!</code>	부정 연산자	<code>!var</code>	참은 거짓, 반대로 거짓은 참으로 바꾼다.
<code>++</code>	증기 연산자	<code>++var, var++</code>	변수 <code>var</code> 의 값을 1 증가한다.
<code>--</code>	감소 연산자	<code>--var, var--</code>	변수 <code>var</code> 의 값을 1 감소한다.

## ● 이항 연산자

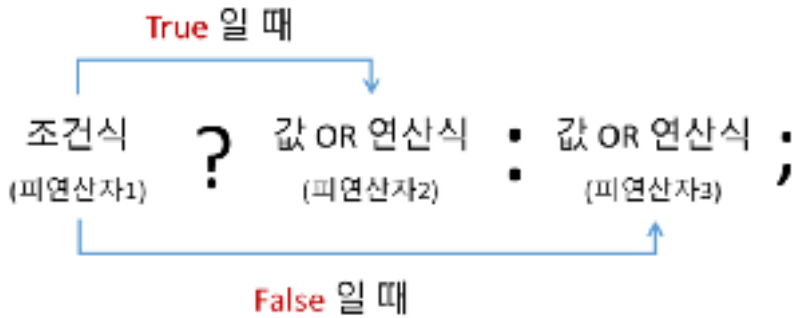
### 산술 연산자

연산식			설명
피연산자	+	피연산자	덧셈 연산
피연산자	-	피연산자	뺄셈 연산
피연산자	*	피연산자	곱셈 연산
피연산자	/	피연산자	좌측 피연산자를 우측 피연산자로 나눴셈 연산
피연산자	%	피연산자	좌측 피연산자를 우측 피연산자로 나눈 나머지를 구하는 연산

### 비교 연산자

구분	연산식			설명
동등 비교	피연산자	==	피연산자	두 피 연산자의 값이 같은지를 검사
	피연산자	!=	피연산자	두 피 연산자의 값이 다른지를 검사
크기 비교	피연산자	>	피연산자	피 연산자 1 이 큰지를 검사
	피연산자	>=	피연산자	피 연산자 1 이 크거나 같은지를 검사
	피연산자	<	피연산자	피 연산자 1 이 작은지를 검사
	피연산자	<=	피연산자	피 연산자 1 이 작거나 같은지를 검사

## 삼항 연산자



## 대입 연산자

구분	연산식			설명
단순 대입 연산자	변수	=	피연산자	우측의 피연산자의 값을 변수에 저장
복합 대입 연산자	변수	+=	피연산자	우측의 피연산자의 값을 변수의 값과 더한 후에 다시 변수에 저장 (변수=변수+피연산자 와 동일)
	변수	-=	피연산자	우측의 피연산자의 값을 변수의 값에서 뺀 후에 다시 변수에 저장 (변수=변수-피연산자 와 동일)
	변수	*=	피연산자	우측의 피연산자의 값을 변수의 값과 곱한 후에 다시 변수에 저장 (변수=변수*피연산자 와 동일)
	변수	/=	피연산자	우측의 피연산자의 값으로 변수의 값을 나눈 후에 다시 변수에 저장 (변수=변수/피연산자 와 동일)
	변수	%=	피연산자	우측의 피연산자의 값으로 변수의 값을 나눈 후에 나머지를 변수에 저장 (변수=변수%피연산자 와 동일)
	변수	&=	피연산자	우측의 피연산자의 값과 변수의 값을 & 연산 후 결과를 변수에 저장 (변수=변수&피연산자 와 동일)
	변수	=	피연산자	우측의 피연산자의 값과 변수의 값을   연산 후 결과를 변수에 저장 (변수=변수 피연산자 와 동일)

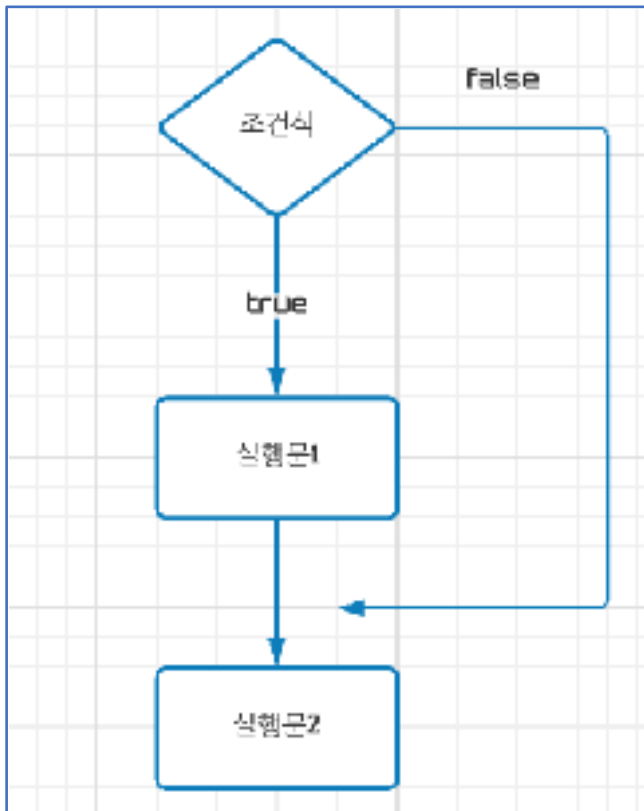
## 논리 연산자

구분	연산식			결과	설명
AND (논리곱)	true	&& 또는 &	true	true	피연산자 모두가 true일 경우만 연산 결과는 true
	true		false	false	
	false		true	false	
	false		false	false	
OR (논리합)	true	 또는 	true	true	피연산자중 하나만 true이면 연산 결과는 true
	true		false	true	
	false		true	true	
	false		false	false	
XOR (배타적 논리합)	true	^	true	false	피 연산자가 하나는 true 이고 다른 하나가 false 일 경우에만 연산 결과는 true
	true		false	true	
	false		true	true	
	false		false	false	
NOT (논리부정)		!	true	false	피 연산자의 논리값을 바꿈
			false	true	

# 조건문

- 조건식에 따라 다른 명령문을 실행하기 위해서 사용한다.

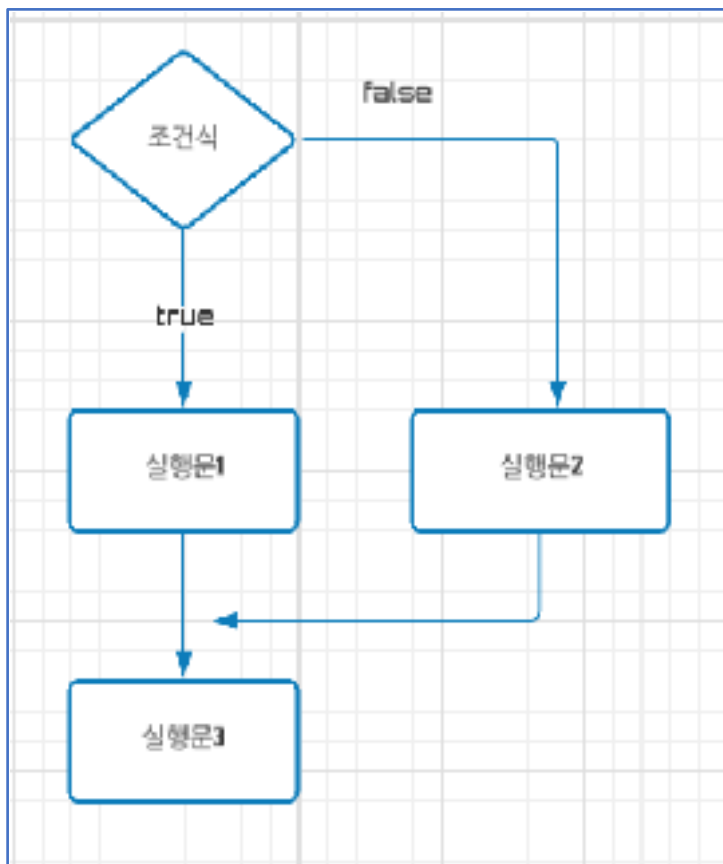
## 1. if 문



```
if(조건식) {  
    실행문1;  
}  
실행문2;
```

순서도 (flowchart)

## 2. if-else 문 (양자 택일)

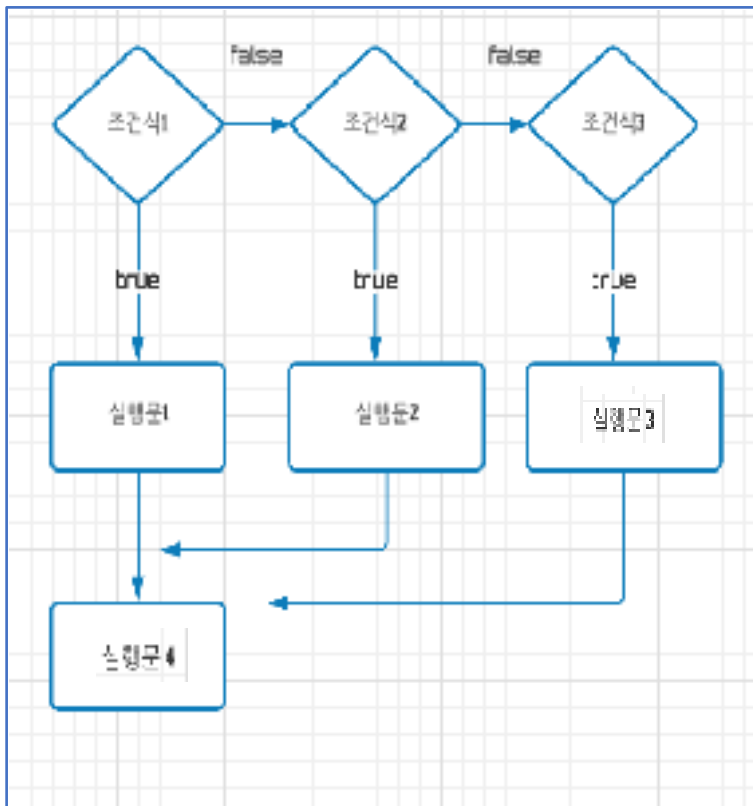


```

if(조건식) {
    실행문1;
} else {
    실행문2;
}
실행문3;
  
```



### 3. 다중 if-else 문 (다중선택)

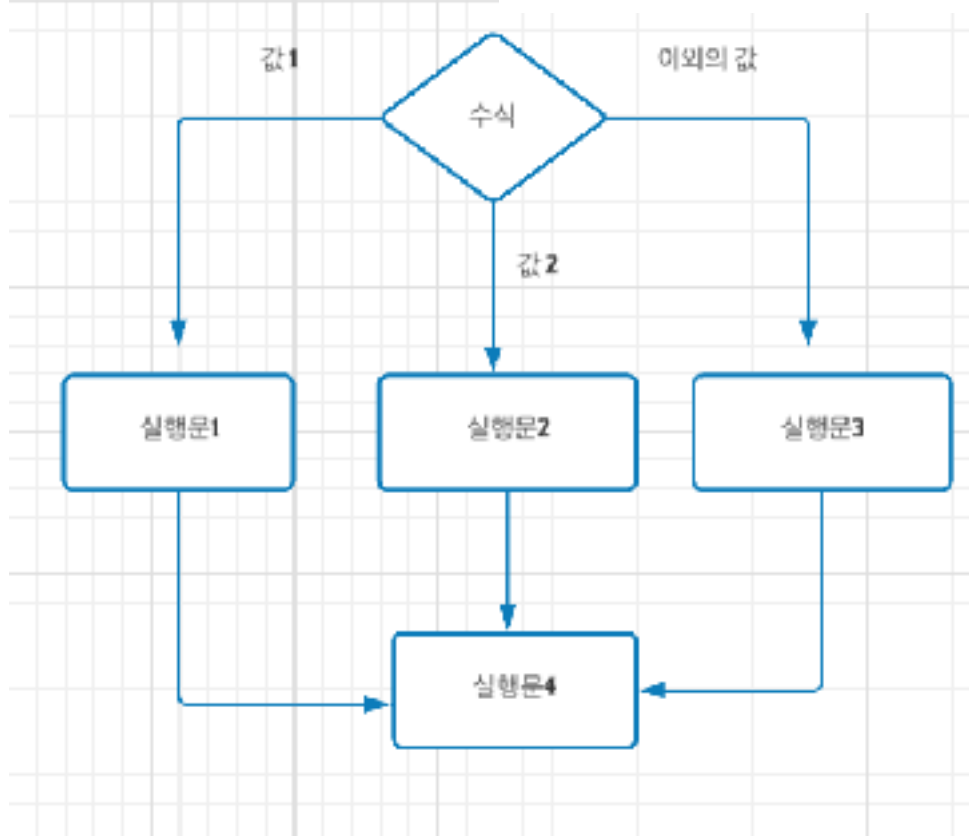


```

if(조건식1) {
    실행문1;
} else if (조건식2) {
    실행문2;
} else if(조건식3) {
    실행문3;
}
실행문4;
  
```

#### 4. swich-case 문

※ 수식은 정수형, 문자열형, 열거형만 올 수 있다.

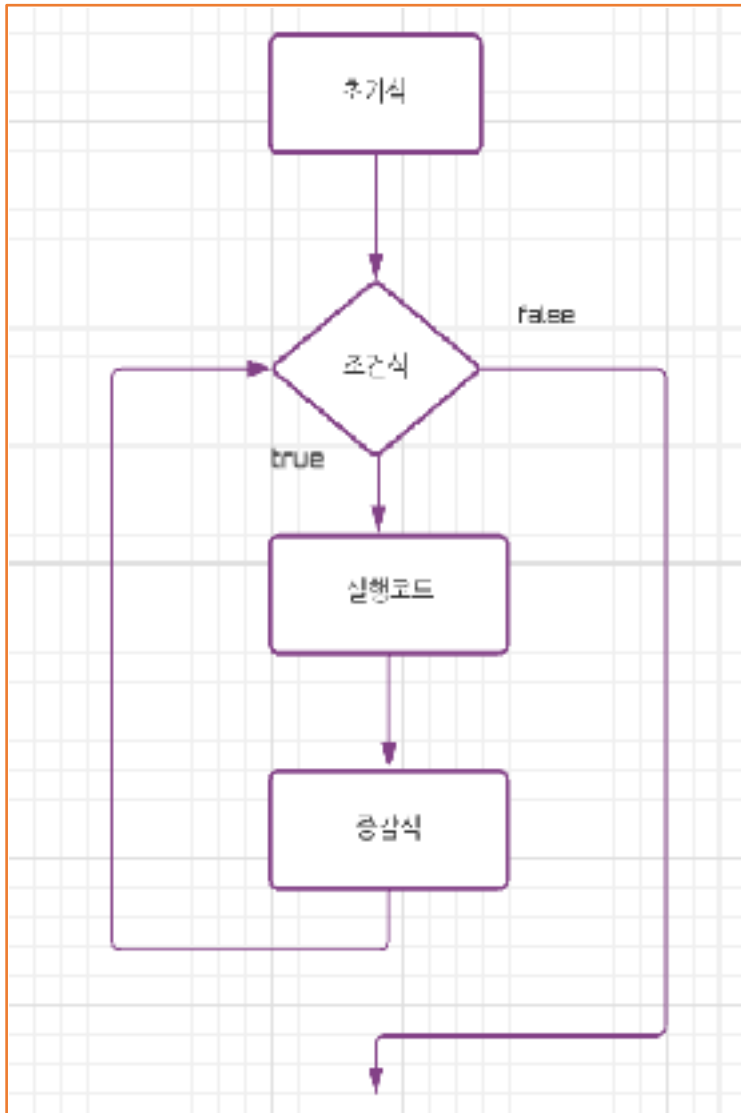


```

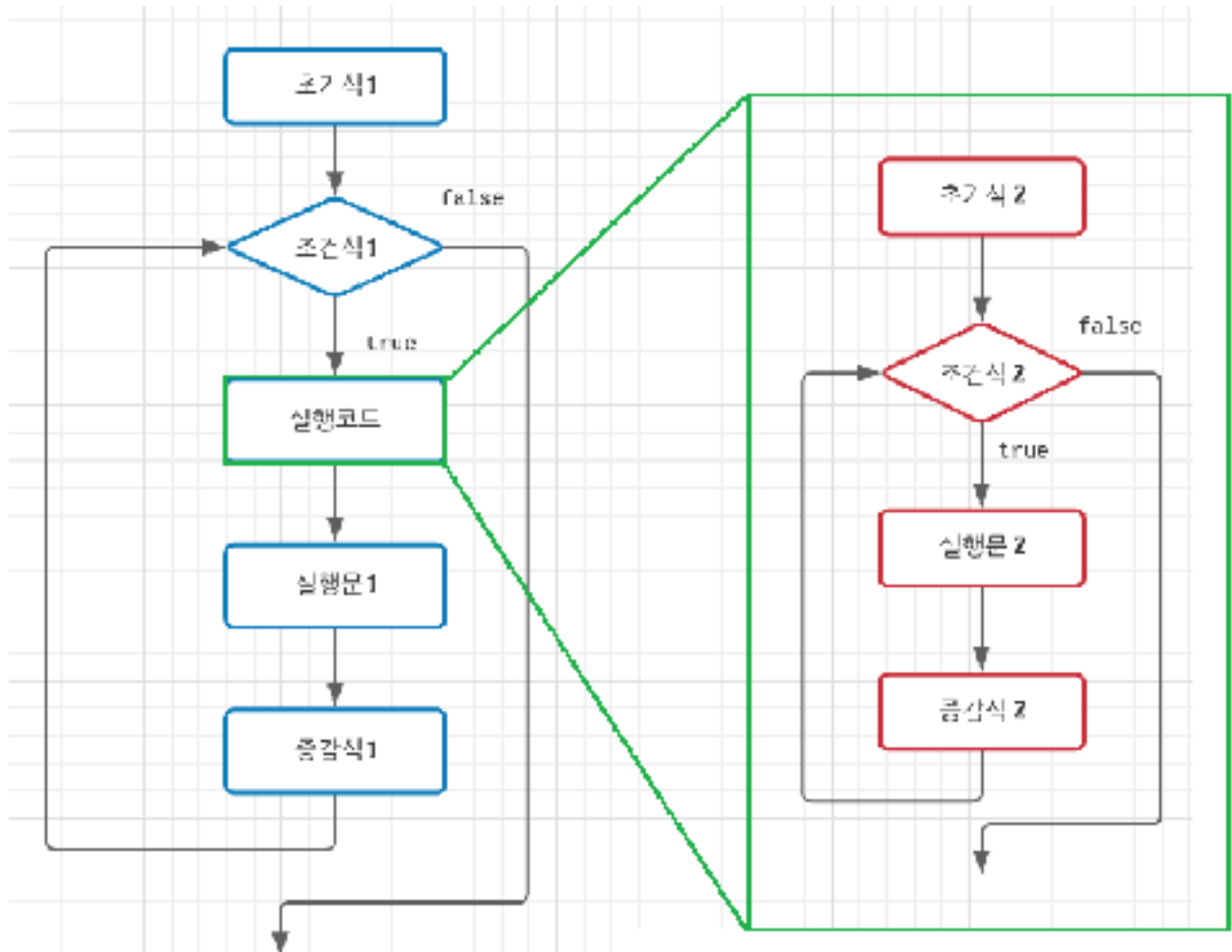
switch( 수식 ) {
    case 값1: 실행문1; break;
    case 값2: 실행문2; break;
    default: 실행문3;
}
실행문4;
  
```

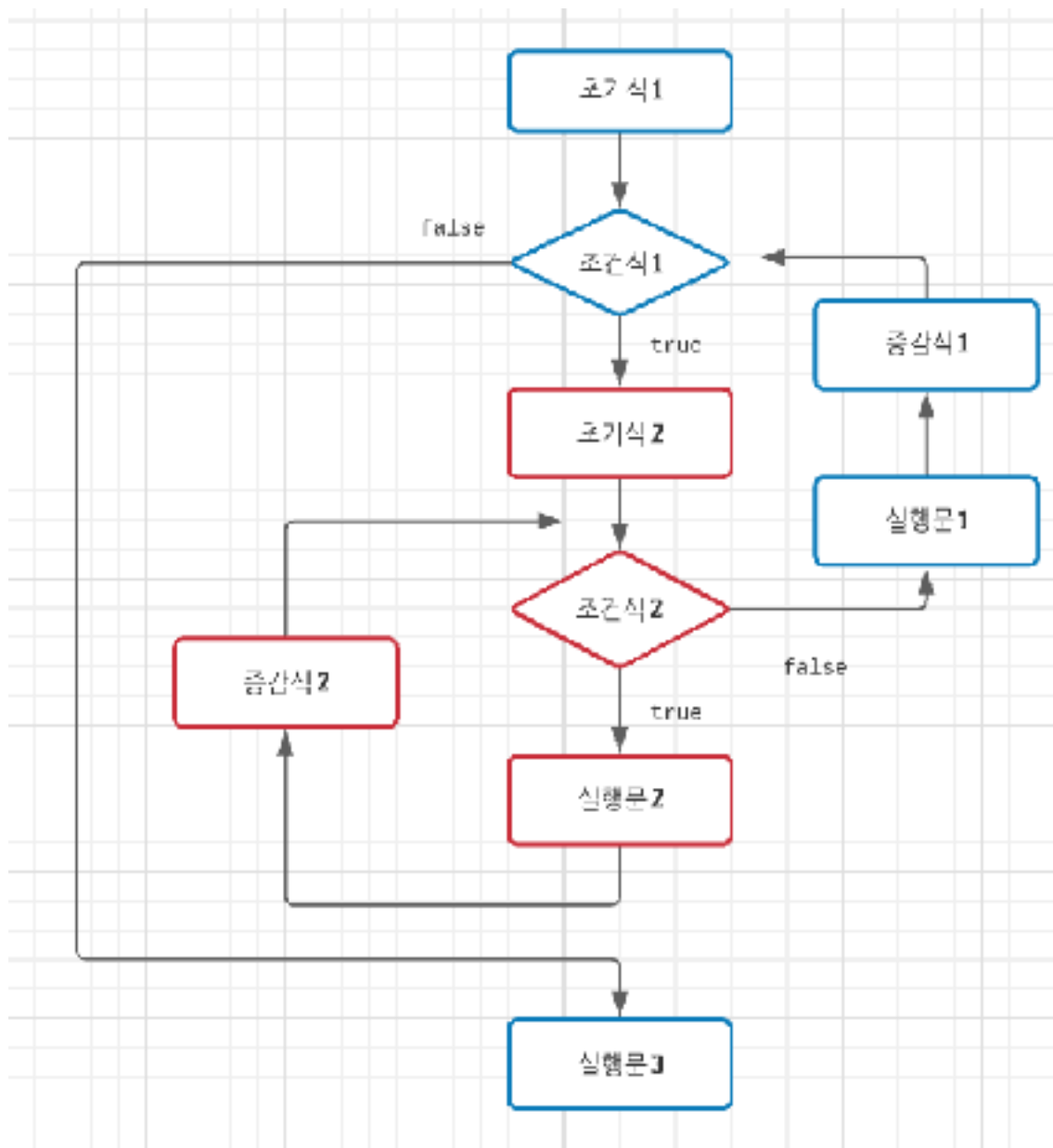
# 반복문

for 문



## 다중 for 문





## break 문

- 반복문, switch 문 또는 label 문을 종료하고, 그 다음 문으로 프로그램 제어를 넘긴다.

[구문]

```
break [ label ];
```

## continue 문

- 반복문 ( while, do-while, for )을 다시 시작하기 위해 사용된다.
- continue 문을 사용하는 경우 가장 안쪽의 while, do-while, for 문을 둘러싼 현재 반복을 종료하고, 다음 반복으로 루프의 실행을 계속한다. break 문과 달리, continue 문은 전체 루프의 실행을 종료하지 않는다. while 루프에서 그것은 다시 조건으로 이동하고, for 루프에서 그것은 증가 표현으로 이동한다.

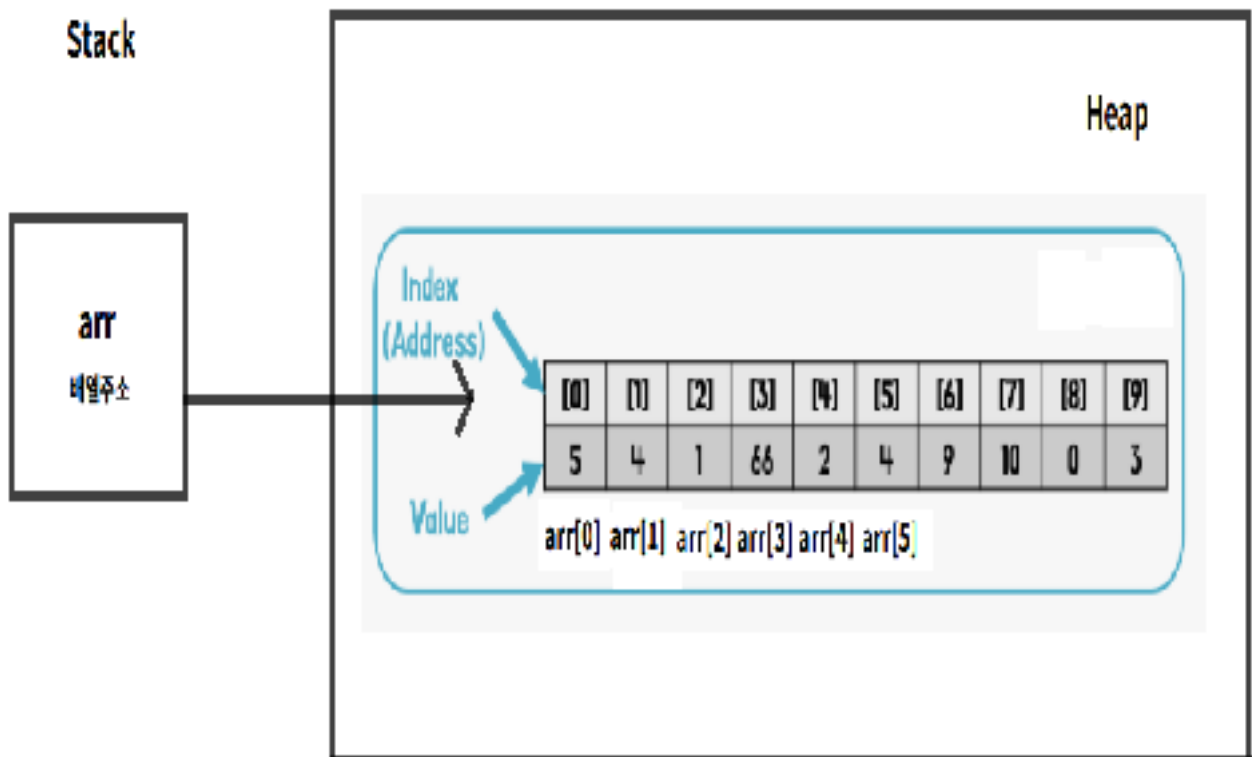
[구문]

```
continue;
```

## 배열 (Array)

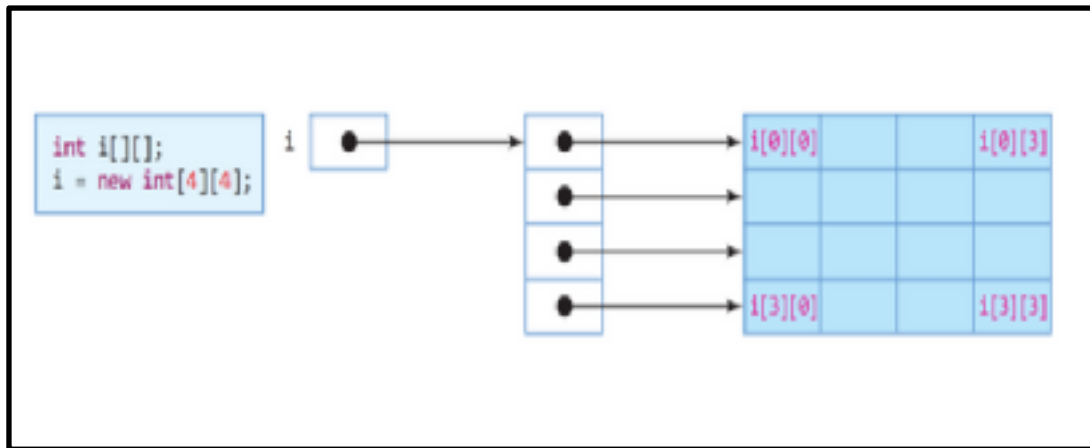
- 번호(인덱스)와 번호에 대응하는 데이터들로 이루어진 자료 구조 를 나타낸다. ·
- 일반적으로 배열에는 같은 종류의 데이터들이 순차적으로 저장되어, 값의 번호가 곧 배열의 시작점으로부터 값이 저장되어 있는 상대적인 위치가 된다.
- 자바에서는 배열의 길이를 구하기 위해서 length 속성을 사용한다.

```
int[] arr = {5, 4, 1, 66, 2, 4, 9, 10, 0, 3}
```

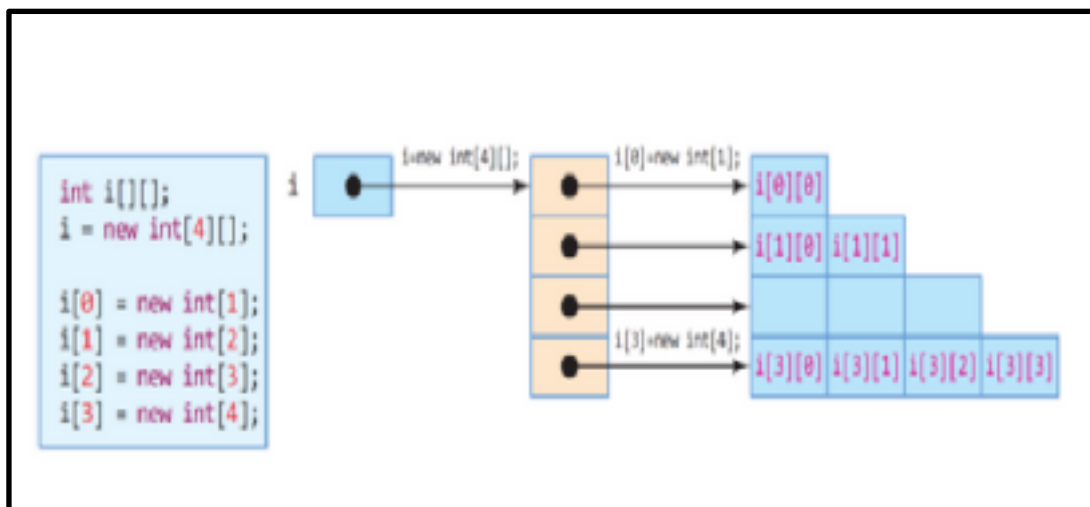


## 다차원 배열

### 정방형 배열



### 비정방형 배열





## for each 구문 (Enhanced for 문)

[구문]

```
for ( 자료형 변수 : 배열이름 ) {
```

```
    실행문;
```

```
}
```

· 배열의 길이 만큼 실행문을 반복하는데 반복이 이루어질 때마나 배열 원소를 순서대로 가져와 변수에 자동으로 대입해준다.

## 메소드

- 중복되는 코드의 반복적인 프로그래밍을 피할 수 있고, 모듈화로 인해 코드의 가독성도 좋아진다.
- 메소드를 작성할 때는 되도록 하나의 메소드가 하나의 기능만을 수행하도록 작성하는 것이 좋다. (응집도가 높다.)

### ● 메소드 선언

```
접근제어자 [static] 리턴타입 메소드명 ( 자료형 변수명, ... ) { // 매개변수목록  
  
    실행코드  
  
    return 값; // return  
  
}
```

1. 접근 제어자 : 해당 메소드에 접근할 수 있는 범위를 명시한다.
2. 리턴타입 : 메소드가 모든 작업을 마치고 반환하는 자료형을 명시한다.  
  
 단, 리턴값이 없는 경우 void 를 지정한다.
3. 메소드 이름 : 메소드를 호출하기 위한 이름을 명시한다. (첫글자는 소문자를 원칙으로한다.)
4. 매개변수 목록 ([parameterlist](#)) : 메소드 호출 시에 전달되는 인수의 값을 저장할 변수들을 명시한다.

## ● 메소드 호출

메소드명( ); // 매개변수가 없는 메소드 호출

메소드명( 값 1, 값 2, ...); // Arguments List ( 인자 목록)