

Java Programming

Java Prepare
Java Syntax
Variables & Operator

01

Java Prepare

01. Java Prepare

자바 개발환경 준비

- JDK15 설치

구글 접속 후 JDK15 검색 ► URL 접속 ► Windows x64 Installer 다운로드
► 다운로드 받은 파일을 더블클릭 하여 설치

The image shows a Google search for 'jdk15' on the left and the Oracle JDK15 download page on the right. The Google search results show a link to 'Java SE 15 Archive Downloads - Oracle' which is highlighted with a red box. The Oracle page shows a table of download links for various operating systems, with the 'Windows x64 Installer' link also highlighted with a red box.

Platform	Size	Download Link
Linux x64 RPM Package	162.03 MB	jdk-15.0.2_linux-x64_bin.rpm
Linux x64 Compressed Archive	179.35 MB	jdk-15.0.2_linux-x64_bin.tar.gz
macOS Installer	175.93 MB	jdk-15.0.2_osx-x64_bin.dmg
macOS Compressed Archive	176.51 MB	jdk-15.0.2_osx-x64_bin.tar.gz
Windows x64 Installer	159.71 MB	jdk-15.0.2_windows-x64_bin.exe
Windows x64 Compressed Archive	179.28 MB	jdk-15.0.2_windows-x64_bin.zip

01. Java Prepare

자바 개발환경 준비

- 환경변수 설정

제어판 ► 시스템 ► 고급시스템 설정 ► 환경변수 JAVA_HOME 과 PATH 설정

정보

PC가 모니터링되고 보호됩니다.

자세한 내용은 Windows 보안을 참조하세요.

장치 사양

장치 이름 DESKTOP-HMNQNFQ
프로세서 AMD Ryzen 5 5600X 6-Core Processor
3.70 GHz
설치된 RAM 64.0GB
장치 ID
제품 ID
시스템 종류 64비트 운영 체제, x64 기반 프로세서
팬 및 터치 팬 지원

복사

이 PC의 이름 바꾸기

관련 설정

[BitLocker 설정](#)

[장치 관리자](#)

[원격 데스크톱](#)

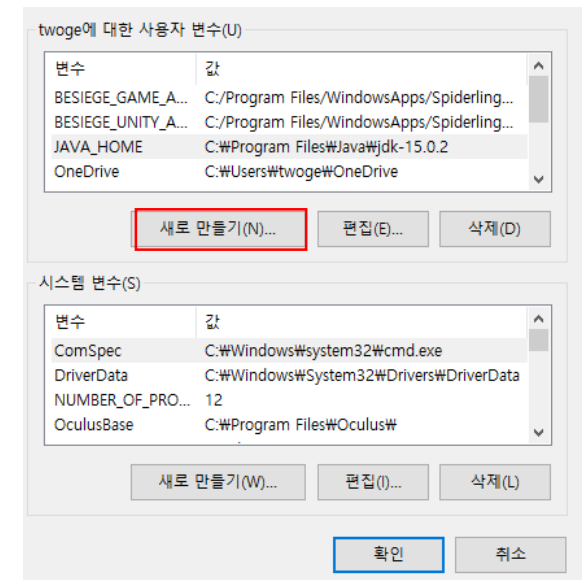
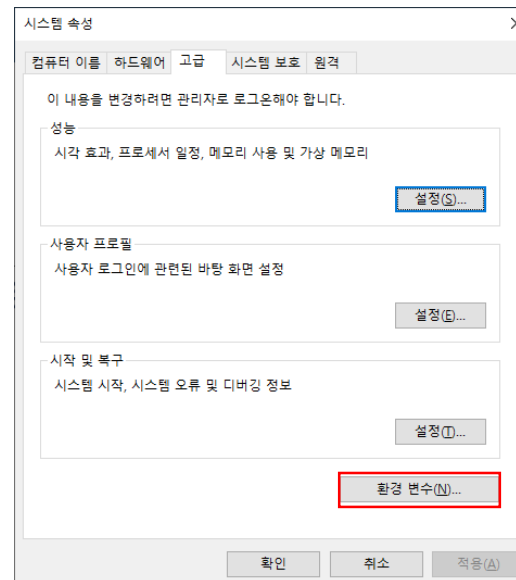
시스템 보호

[고급 시스템 설정](#)

[이 PC의 이름 바꾸기\(고급\)](#)

[도움말 보기](#)

[피드백 보내기](#)

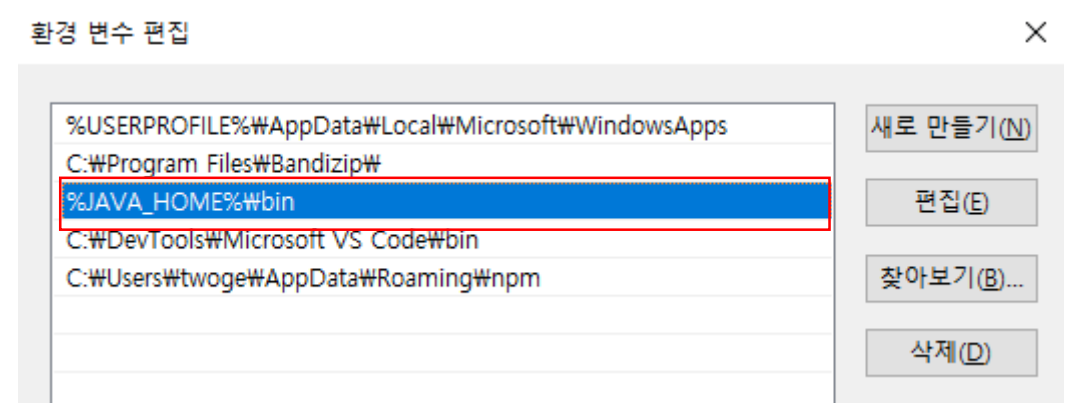
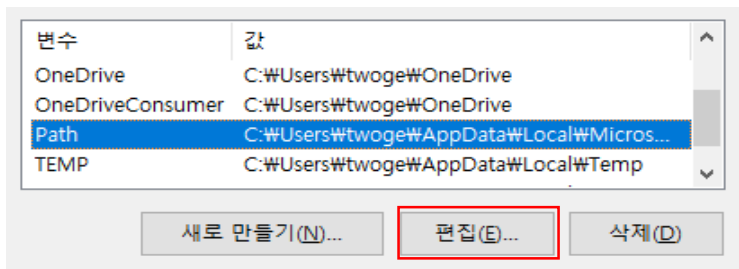
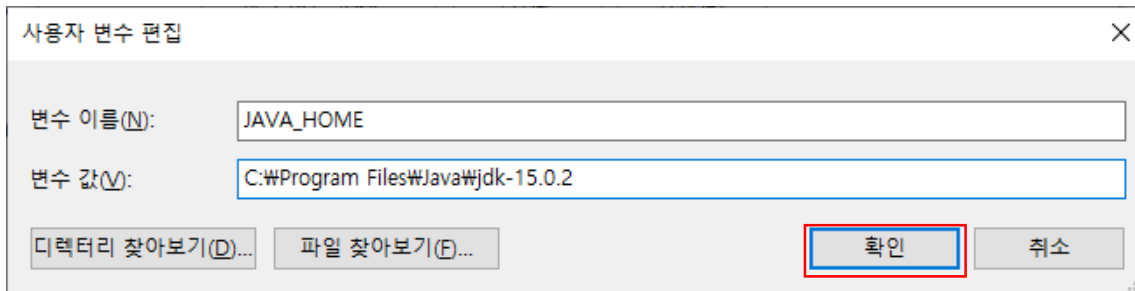


01. Java Prepare

자바 개발환경 준비

- 환경변수 설정

제어판 ► 시스템 ► 고급시스템 설정 ► 환경변수 JAVA_HOME 과 PATH 설정

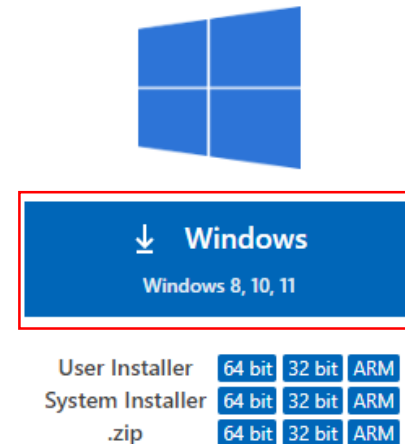
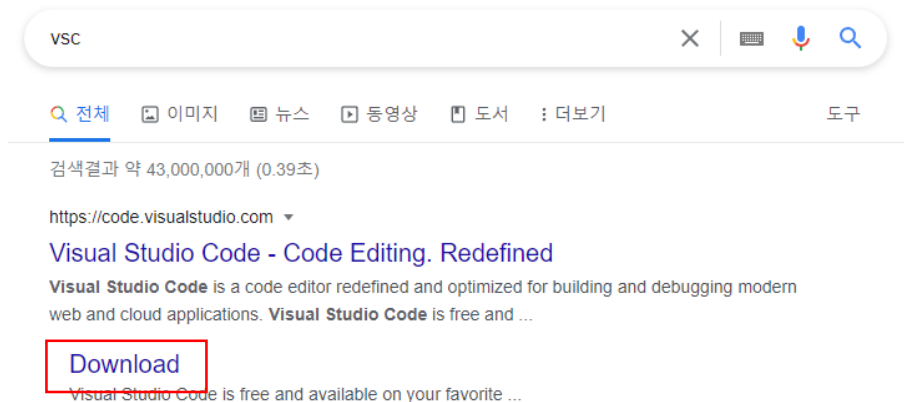


01. Java Prepare

자바 개발환경 준비

- IDE (Integrated Development Environment) 설치

Google 접속 후 vsc 검색 ► Download URL 접속 ► 다운로드 ► 실행

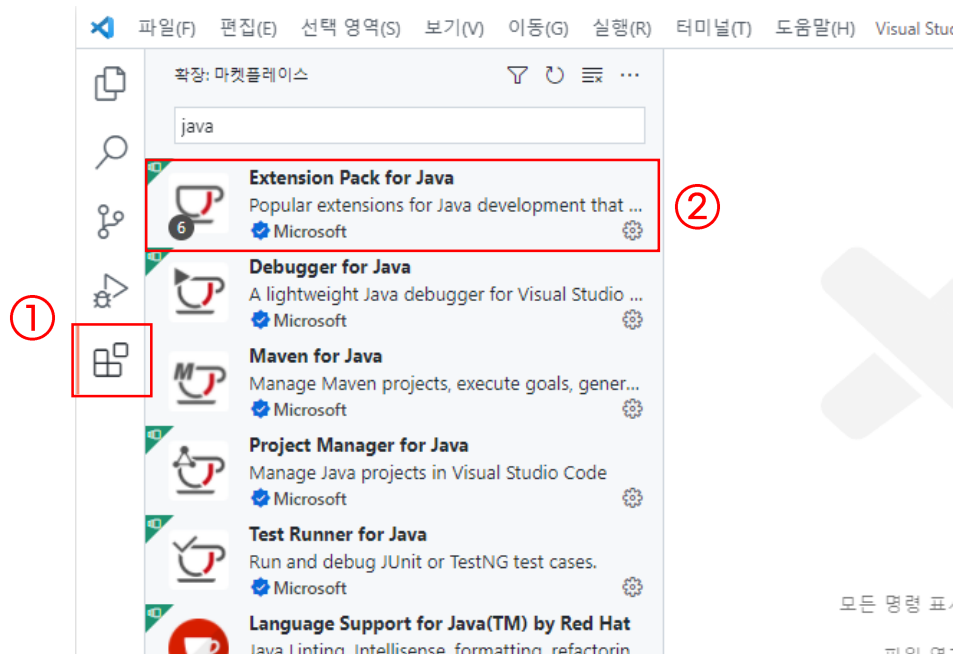


01. Java Prepare

자바 개발환경 준비

- Visual Studio Code 추가기능 설치

VSCoDe 실행 ► 확장 관리 ► Extension Pack for Java 설치

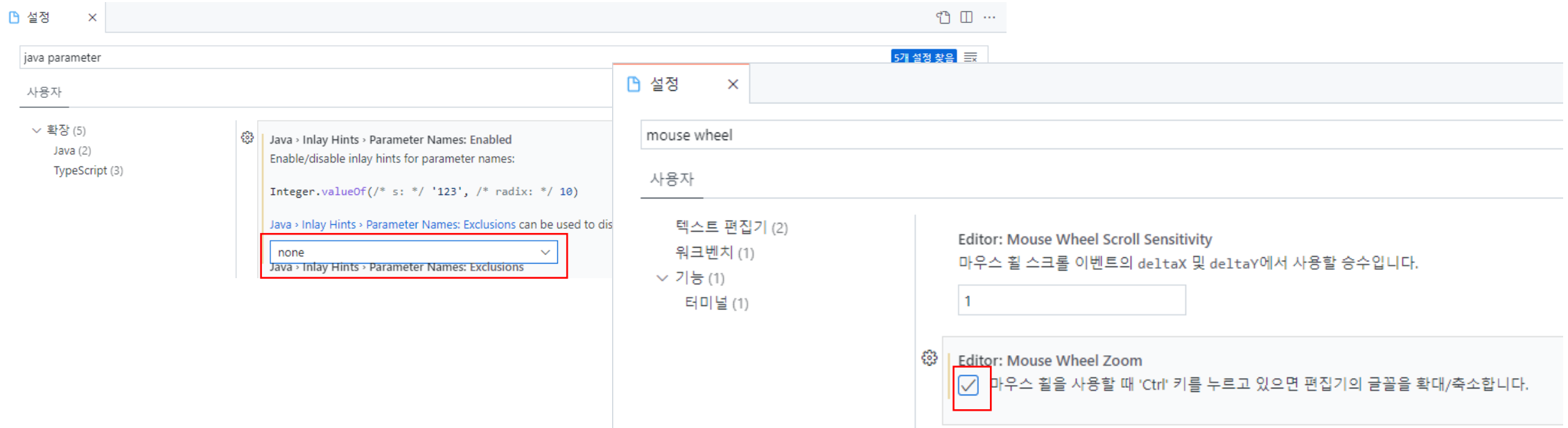


01. Java Prepare

자바 개발환경 준비

- Visual Studio Code 기능 설정

VSCoDe 실행 ► 파일 ► 기본설정 ► 설정

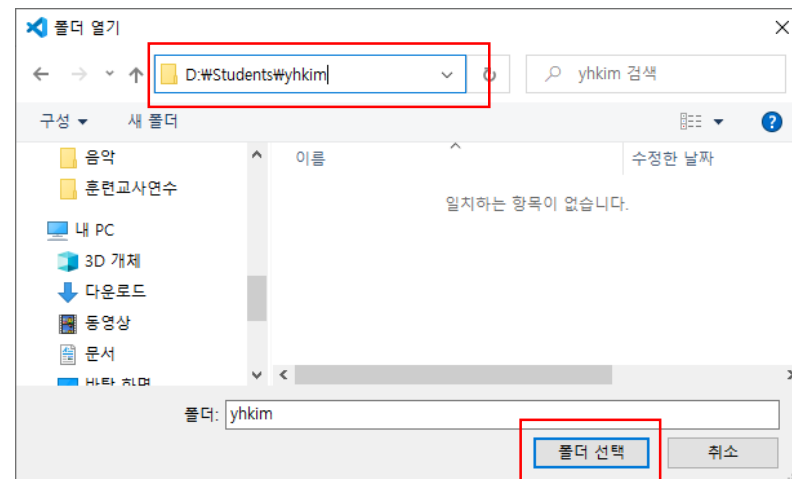


01. Java Prepare

자바 개발환경 준비

- 샘플 자바 프로그램 실행

파일 ► 폴더열기 ► 작업폴더 선택 ► 탐색기 ► 우클릭 ► 새파일 ► First.java 입력

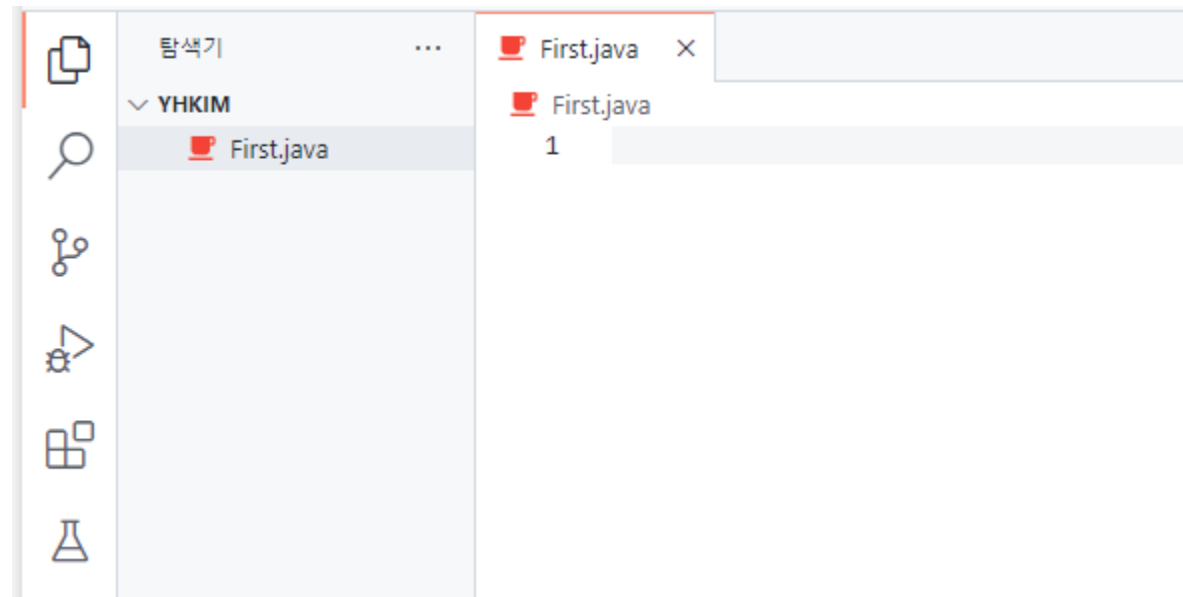
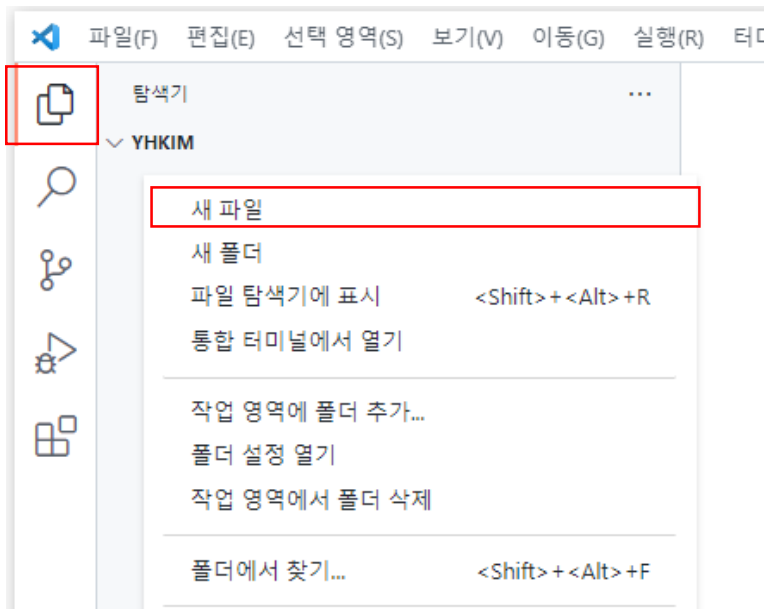


01. Java Prepare

자바 개발환경 준비

- 샘플 자바 프로그램 실행

파일 ► 폴더열기 ► 작업폴더 선택 ► 탐색기 ► 우클릭 ► 새파일 ► First.java 입력



01. Java Prepare

자바 개발환경 준비

- 샘플 자바 프로그램 실행

활성화 된 편집창에 코드 입력 ► Ctrl + F5 단축키로 실행

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



02

Java Syntax

02. Java Syntax - First.java

First.java 분석

- 자바 프로그램 코드는 블록 단위로 만들어진다.
- 블록 시작기호는 { 로 쓰며 끝 기호는 } 로 쓴다.
- 블록기호는 시작과 끝 기호가 쌍이 맞아야 한다.

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

02. Java Syntax - First.java

First.java 분석

- 자바 프로그램 코드는 블록 단위로 만들어진다.
- 블록 시작기호는 { 로 쓰며 끝 기호는 } 로 쓴다.
- 블록기호는 시작과 끝 기호가 쌍이 맞아야 한다.
- 자바 파일 내부에 main은 단 하나만 존재한다.
- 일반 괄호 () 는 파라미터(Parameter)를 위한 공간이며, 블록이 아니다.
- 대괄호 [] 는 배열(Array) 타입을 위한 기호이며, 블록이 아니다.

02. Java Syntax - First.java

First.java 분석

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

First class의 시작 기호

First class의 내용 부분

First class의 끝 기호

02. Java Syntax - First.java

First.java 분석

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

main의 시작 기호

main의 내용 부분

main의 끝 기호

The diagram illustrates the structure of the `main` method in the `First.java` file. It shows the following code snippet:

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

Annotations:

- A red arrow points to the opening curly brace `{` of the `main` method, labeled "main의 시작 기호" (Start symbol of main).
- A red double-headed arrow indicates the content of the `main` method, labeled "main의 내용 부분" (Content part of main).
- A red arrow points to the closing curly brace `}` of the `main` method, labeled "main의 끝 기호" (End symbol of main).

02. Java Syntax - First.java

First.java 분석

- System.out.println(); 은 기능을 실행하는 명령문이며 세미콜론(;) 으로 끝난다.
- 기초를 학습하는 단계에서는 명령문은 반드시 main 블록 안쪽에 위치한다.

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

명령문

02. Java Syntax - First.java

First.java 분석

- 명령문 사이사이에 있는 dot (.) 은 ~의 혹은 ~안에 있는 으로 해석한다.

```
System.out.println("Hello World!");
```

System 안에 있는 out 안에 있는 println 을 실행시키는데
입력 값으로 "Hello World!" 를 전달한다.

02. Java Syntax - First.java

First.java 분석

System

- Java 의 내장 클래스 System

out

- Java 의 표준 출력 (Console화면)

`println("Hello World!");`

- 괄호 안의 내용을 한 줄로 표시하는 기능(Function) 혹은 방법(Method)

02. Java Syntax - Exercise 01

연습문제 01

- 클래스 명을 UserInfo 로 한다.
- 사용자의 아이디, 이메일, 가입일을 콘솔에 표시하는 프로그램을
제시된 예시를 참고하여 작성한다.

```
아이디 : twogenesis  
이메일 : twogenesis@naver.com  
가입일 : 2022-10-17
```

03

Java Variables&Operators

03. Java Variables

변수

- 변수의 본래 뜻은 변할 가능성이 있는 수를 의미한다.
- 컴퓨터 시스템에서의 변수는 값을 저장하기 위한 메모리공간이라는 뜻이 추가된다.
- 변수는 저장할 데이터의 형태와 크기에 따른 "자료형"(Data Type) 이 있다.

03. Java Variables

변수의 자료형 (Data types)

기본 자료형 (Primitive Data Types)

- 정수형, 실수형, 문자형, 논리형이 있다.
- 자바 시스템에서 미리 정해 둔 자료 형, 기본 값이 있음.

참조형 (Reference Types)

- 기본형을 제외한 타입들은 모두 참조형 타입
- 현 단계에서 가장 대표적인 참조형은 String 타입
- 기본 값은 비어있음을 의미하는 null 이다.

03. Java Variables

기본 자료형 (Primitive Data Types)

정수형 (Integer Types)

- 메모리 공간의 크기에 따라 byte, short, int, long 타입이 있다.
- int 타입이 연산속도가 가장 빠르기 때문에 통상적으로 int 타입을 사용한다.

실수형 (Floating Point Types)

- 메모리 공간의 크기에 따라 float, double이 있다.
- double타입이 정밀도가 더 좋기 때문에 통상적으로 double 타입을 사용한다.

03. Java Variables

Bit & Byte

- 1Byte는 8Bit로 구성되며, 1Bit는 2진수의 자리 수 1자리를 의미한다.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	0	0	0	0
1Byte							

- 1Byte로 표현할 수 있는 숫자의 개수는 $128+64+32+16+8+4+2+1 = 255$ 개 이다.

03. Java Variables

기본 자료형 (Primitive Data Types)

크기	1Byte	2Bytes	4Bytes	8Bytes
정수형	byte	short	기본 int	long
실수형			float	기본 double
문자형		char		
논리형	boolean			

03. Java Variables

변수의 생성(정의)

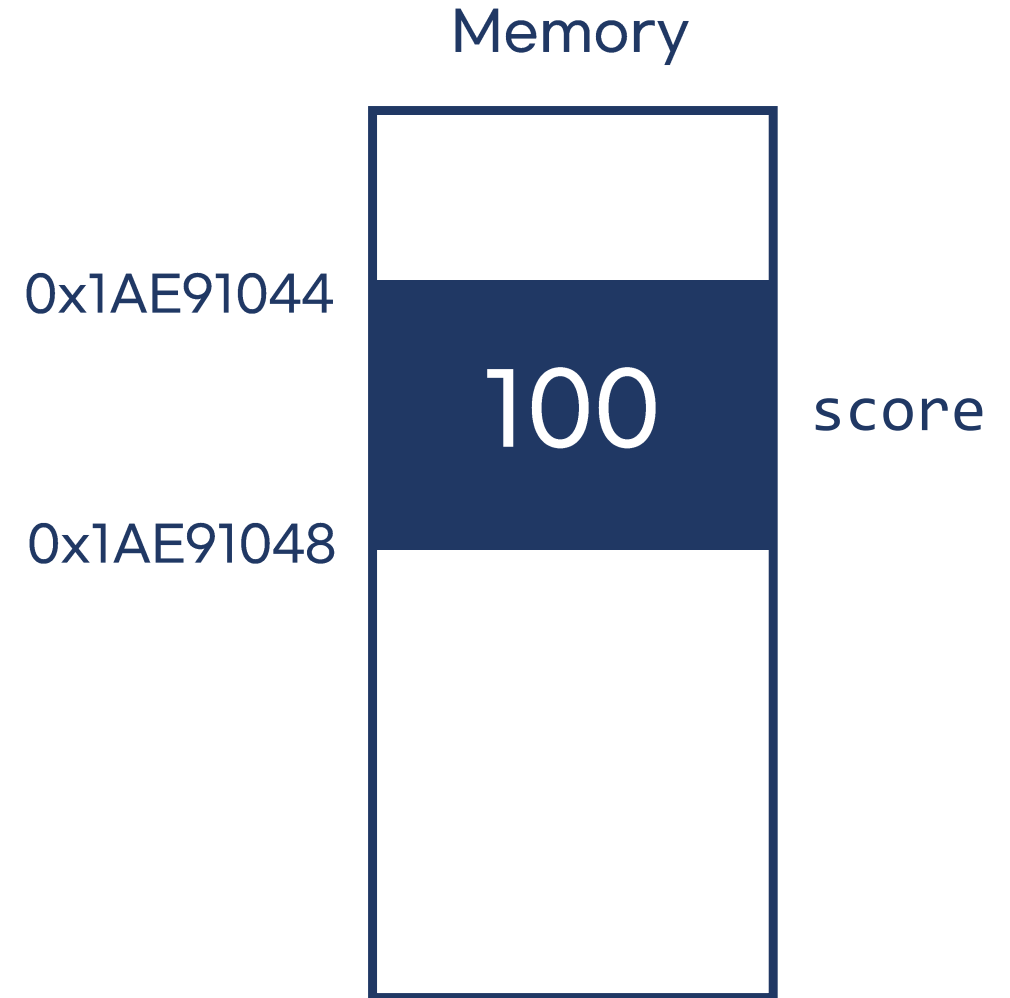
변수자료형 변수이름(식별자);

```
int score;
```

변수의 초기화

변수이름(식별자) = 값;

```
score = 100;
```



03. Java Variables

정수형 변수 타입

`byte value1 = 127;`

-128 ~ 127

`short value2 = 32235;`

-32768 ~ 32767

`int value3 = 100;`

-2147483648 ~ 2147483647

`long value4 = 100000000000L;`

-9223372036854775808 ~ 9223372036854775807

03. Java Variables

실수형 변수 타입

```
float f1 = 1.23f;
```

$(3.4 \times 10^{-38}) \sim (3.4 \times 10^{38})$ 의 근사값

```
double d1 = 1.23456;
```

$(1.7 \times 10^{-308}) \sim (1.7 \times 10^{308})$ 의 근사값

문자형 변수 타입

```
char c = 'A';
```

0 ~ 65,535

03. Java Variables

논리형 변수 타입

```
boolean b = true;
```

true or false

문자열 타입

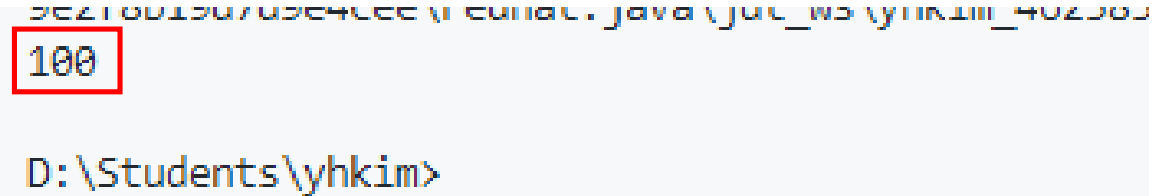
```
String str1 = "Hello";  
String str2 = new String("World!");
```

03. Java Variables

변수의 사용

- 만들어진 변수의 식별자를 println 으로 전달

```
public class VariablesEx {  
    public static void main(String[] args) {  
        int i = 100;  
        System.out.println(i);  
    }  
}
```



```
3e2100130702e40ee \main.java \jdk_ws \ytkim_402303  
100  
D:\Students\yhkim>
```

03. Java Variables

변수의 값 변경

- 이미 만들어진 변수는 재사용 시 변수 타입을 쓰지 않는다.

```
public class VariablesEx {  
    public static void main(String[] args) {  
        int i = 100;  
        System.out.println(i);  
        i = 200;  
        System.out.println(i);  
    }  
}
```



```
100  
200  
D:\Students\yhkim>
```


03. Java Variables - Exercise 02

연습문제

다음에 제시된 값들을 저장할 수 있는 변수를 생성하고, 결과를 확인하시오.

```
123456789012, 22234, 55, 12.34,  
12.3333222, "Hello World", '$'
```