

개발준비

- ① 프로그래밍 기본 용어
- ② 인텔리제이 설치
- ③ 프로젝트 설정 · Hello Java



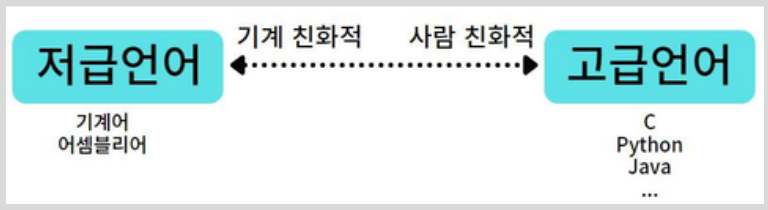
기계어는 CPU가 직접 해독하고 실행할 수 있는 비트 단위로 쓰인 컴퓨터 언어를 통틀어 일컫는다. 기계어는 프로그램을 나타내는 가장 낮은 단계의 개념

기본 용어

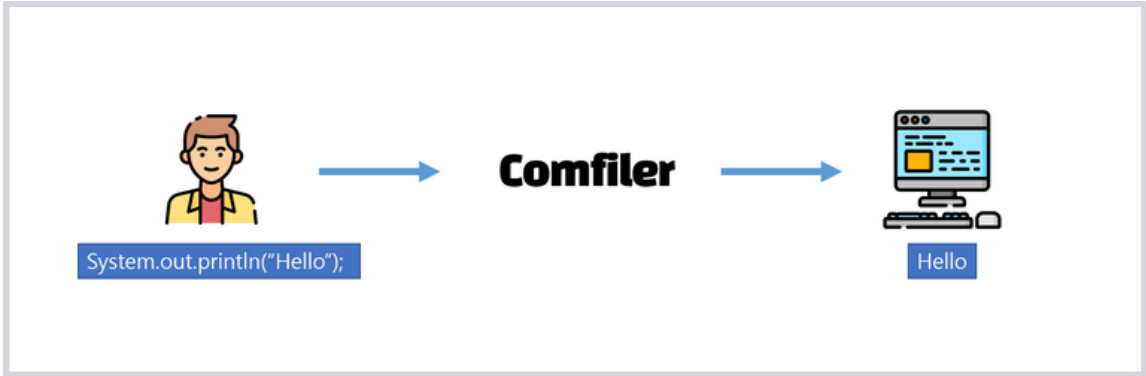
- 기계어 : 컴퓨터가 이해 할 수 있는 언어
 - 2진코드 , 0또는1

```
001001 11101 11101 111111111111000
001000 00001 00000 0000000000001010
001000 00001 00001 0000000000000010
101011 11101 00001 0000000000000000
001000 00010 00001 00000000000000100
101011 11101 00010 00000000000000100
001001 11101 11101 0000000000001000
```

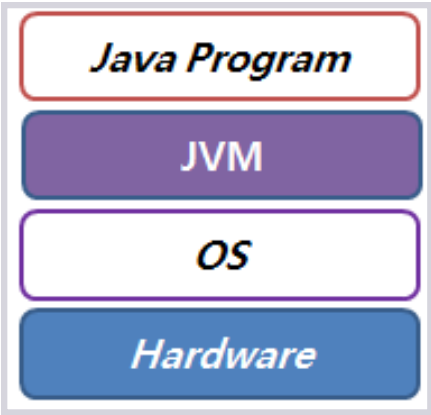
- 고급언어 : 사람이 이해 할 수있는 언어.
 - C , C++ , 자바 , 파이썬 등
- 저급언어 : 컴퓨터가 이해 할 수 있는 언어.
 - 어셈블리어



- 실행 과정
 - 사람(문자/JAVA문법) ----> 소스파일 작성(.java) ---- 컴파일/실행 ----> 기계어파일(.class) ----> 컴퓨터






- 자바 특징
 - ① 모든 운영체제에서 실행 가능
 - 모든 소스코드가 기계어로 변환이 되고 실행 되기 때문에 운영체제로 부터 독립적
 - ② 객체 지향 프로그래밍
 - 부품 만들고 부품들을 연결해서 더 큰 프로그램 완성 ex) 레고
 - ③ 메모리 자동 정리
 - *Garbage Collection (GC) : 사용하지 않는 메모리를 자동으로 초기화/제거
 - ④ 무료 라이브러리 풍부
 - 미리 만들어진 함수/클래스 들을 제공 함으로써 빠른개발/협업 도움 ex) 스프링 프레임워크



- JDK : 자바 개발 도구
 - JAVA문법 과 라이브러리 가지고 있는 코드 집합
 - 버전 : JDK8 , JDK11 , JDK17(강의)
 - 스프링부트3.0 이상부터 JDK17 이상부터 지원 (단* 현재 전자정부프레임워크는 JDK1.8 다수 사용중)
 - JDK17에 JDK8 포함

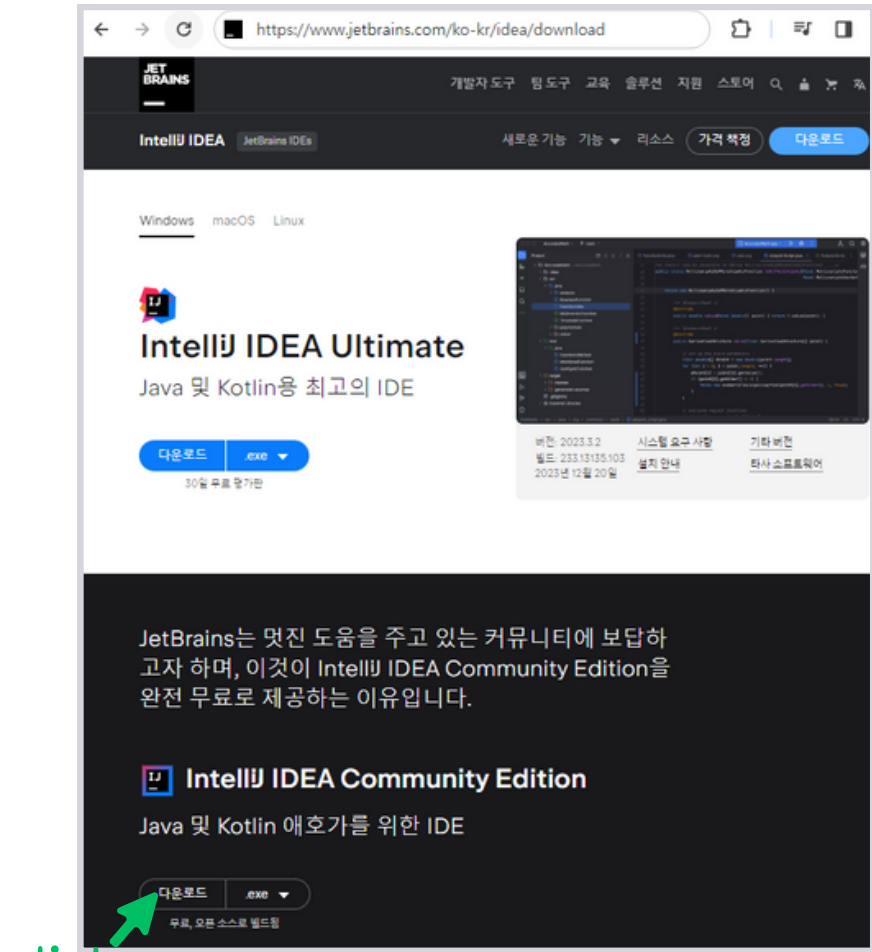
Release	Extended Support Until
8 (LTS)**	December 2030*****
9 - 10 (non-LTS)	Not Available
11 (LTS)	January 2032
12 - 16 (non-LTS)	Not Available
17 (LTS)	September 2029****
18 (non-LTS)	Not Available
19 (non-LTS)	Not Available
20 (non-LTS)	Not Available
21 (LTS)***	September 2031
22 (non-LTS)***	Not Available
23 (non-LTS)***	Not Available
24 (non-LTS)***	Not Available
25 (LTS)***	September 2033****

- 텍스트 에디터
 - 각 언어별 소스 작성시 자동완성 및 개발에 도움을 주는 환경/기능 제공하는 소프트웨어
 - ① Visual Studio Code (프론트엔드 가장 인기) : 프론트엔드 수업 
 - ② IntelliJ IDEA (일반기업/프리랜서 가장 인기) : 백엔드 수업
 - IntelliJ IDEA Ultimate : 유료버전 
 - IntelliJ IDEA Community Edition : 무료 버전
 - ③ Eclipse (백엔드 가장 인기) 

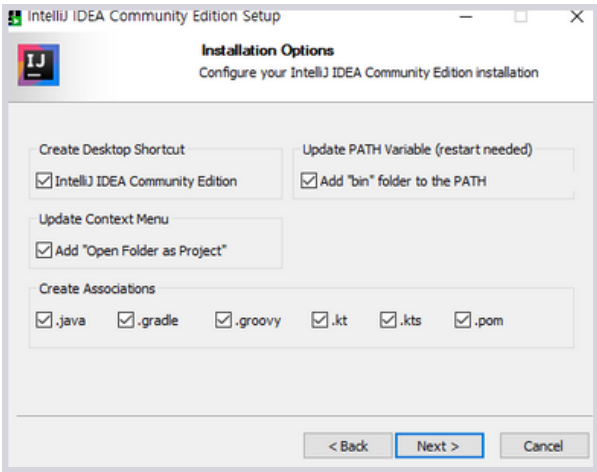


인텔리제이 설치

- 1. 구글 검색창에 ‘인텔리제이’ 검색
https://www.jetbrains.com/ko-kr/idea/
- 2. 인텔리제이 홈페이지서 다운로드 클릭
https://www.jetbrains.com/ko-kr/idea/download
- 3. 인텔리제이 커뮤니티 에디션 다운로드



- 4. 다운로드시 옵션을 모두 체크 해주세요. 필수는 아니지만 권장 합니다.



- 5. 바탕화면에 아이콘을 확인합니다. IntelliJ IDEA Community Edition 이며 버전은 상관이 없습니다.



프로젝트 설정

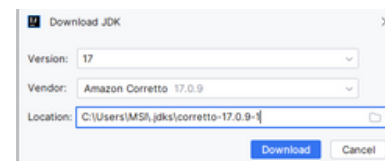
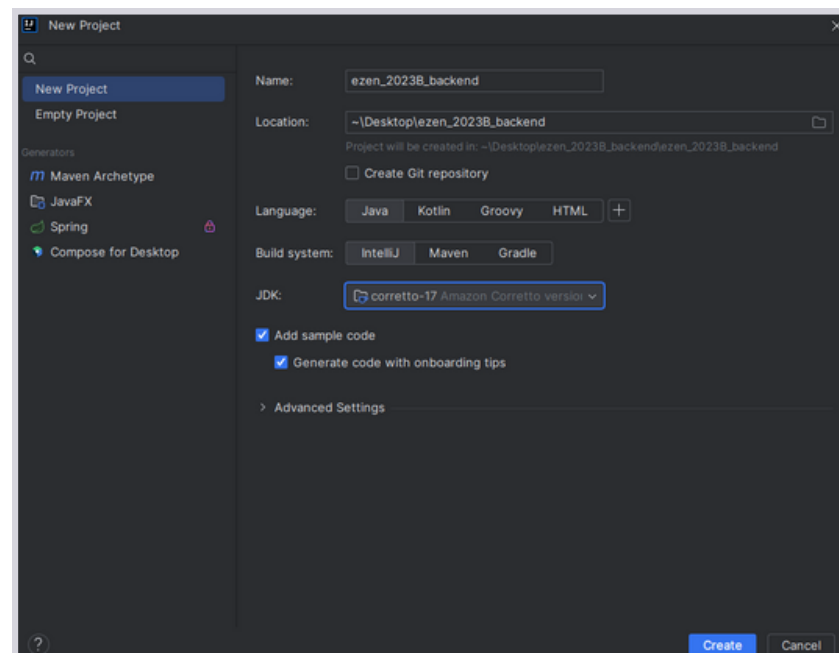
● [IntelliJ IDEA Community Edition 2023.3.2]

1. 실행



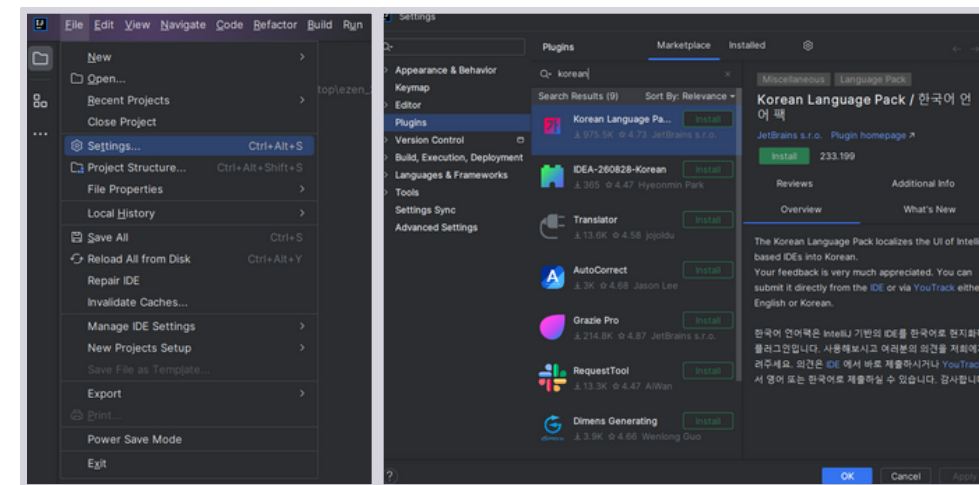
2. New project

- ① name : 프로젝트 이름 ex) ezen_2023B_backend
- ② location : 프로젝트 위치 ex) ~\Desktop\ezen_2023B_backend
- create git repository : 체크 안함.
- ③ Language : java
- ④ build system : intellij
- ⑤ JDK : corretto-17
- PC의 JDK 없을경우 최초 1번 따라하기
 - ① Download jdk
 - ② version : 17* , amazon , C:\Users\(\본인pc이름)\.jdk\corretto-17.0.9(create) 버튼 클릭



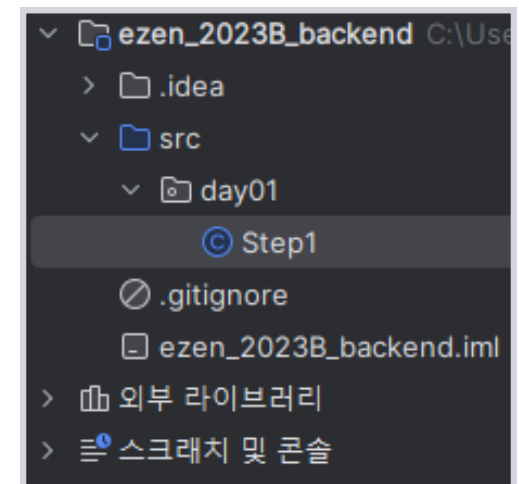
● [인텔리제이 필수 세팅]

- ① 인텔리제이 테마 선택
메뉴 -> 셋팅 -> appearance -> 테마 선택.
- ② 마우스 휠 이용한 확대/축소 기능
메뉴 -> 셋팅 -> editor -> general -> mouse control -> change font size with ctrl+mouse wheel -> active editors
- ③ 플러그인(마켓플레이스)
메뉴 -> 셋팅 -> 플러그인
 1. 'korean' 검색 - 한글패치 [Korean Language Pack]



● [자바 프로젝트 폴더 구성]

- ① 프로젝트이름 [ezen_2023B_backend]
 - idea : 인텔리제이 설정파일
 - out : 컴파일/실행결과 된 파일 (컴퓨터) [.class]
 - src : 컴파일 실행 전 파일/소스파일 (개발자) [.java]
 - ② 폴더/패키지 [해당 폴더 오른쪽클릭 -> 새로만들기 -> 패키지 -> day01
 - ③ 클래스 [해당 폴더 오른쪽클릭 -> 새로만들기 -> 클래스]
Step1 [첫글자는 대문자!!!! (*아닐경우 오류발생)]
 - gitignore : git commit 할때 무시할 파일 등록 하는 곳
 - 외부라이브러리[JDK17]



Hello Java

● 주석

// : 한줄 주석
/* 여러줄 주석 */

● 자바 첫 코딩

– 전체적인 흐름만 경험하며 모든 코드를 이해 할 필요가 없습니다.

Step1 : 자바 시작

```
package day01;

public class Step1 {
    public static void main(String[] args) {
        System.out.println("안녕 자바");
    }
}
```

① 패키지

– 현재 클래스파일의 위치를 표시됩니다. 즉 폴더 개념과 비슷하며 , 식별자 역할 합니다.

코드분석 ① 해당 소스파일은 day01 폴더에 위치한다는 의미 입니다.

```
package day01;
```

② 클래스

- 자바에서 모든 프로그램 소스는 클래스 단위로 시작합니다. 그래서 모든 코드는 클래스 안에서 작성합니다.
- 자세한 내용은 추후에 학습할 예정입니다.
- 클래스명 정의 : 숫자 시작 과 공백을 할 수 없고 일부 특수문자만 가능하며 첫글자가 대문자인 카멜표기법을 권장합니다.

코드분석 ② 클래스 선언하고 정의 합니다.

```
public class Step1 {

}
```

③ main()메소드

- 프로그램 실행하면 main() 메소드 블록이 실행된다. 프로그램의 진입/시작점 이라고도 한다.
- main() 안에 main 스레드(코드를 읽는 흐름) 포함 되므로 무조건 main() 부터 실행이 됩니다. 자세한 스레드는 추후에 학습합니다.
- 자동완성 : m + enter키

코드분석 ③ main() { } 안에 실행하고 싶은 코드를 작성합니다.

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

}
```

④ 콘솔 출력

- 프로그램 실행시 콘솔에 출력 되는 명령어 입니다.
- System.out.println(" ") 소괄호 안에 “ ” 처리 하여 출력하고 싶은 문자열을 작성합니다. 그리고 ; 세미콜론으로 끝 마칩니다.
- 자동완성 : sout + enter키

코드분석 ④ println(); 안에 콘솔에 출력하고 싶은 문자열을 작성합니다.

```
System.out.println("안녕 자바");
```

● 세미콜론

- ; 세미콜론은 컴파일러 에게 명령문이 끝났다는 것의 의미로 전달하여 컴파일러는 해당 명령어를 처리합니다..
- 1.변수선언 2.변수대입 3.메소드호출시 등등 사용됩니다.



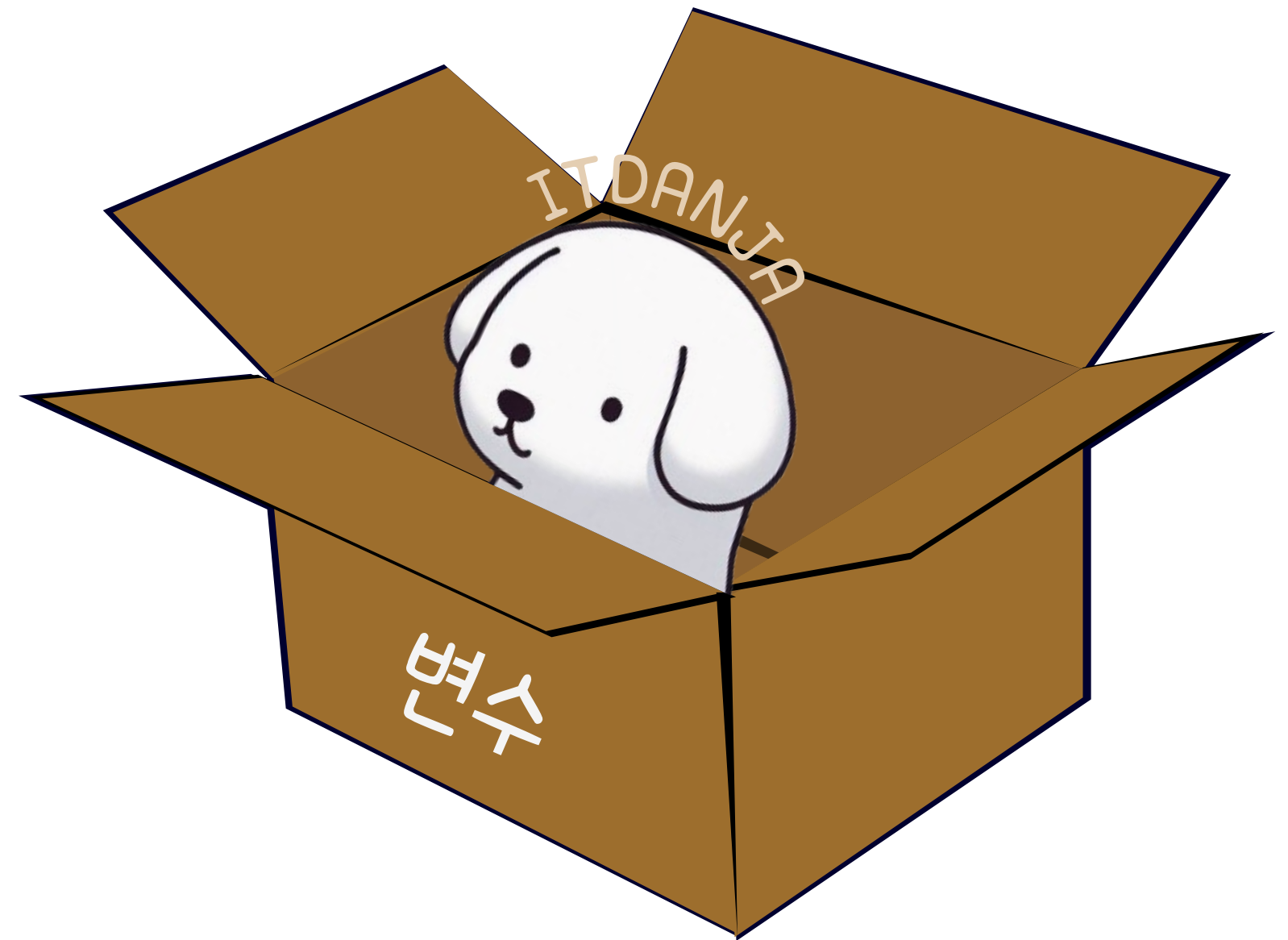
변수

- ① 변수란
- ② 기본 자료형
- ③ 입출력



컴퓨터 프로그래밍에서 변수 또는 스칼라는 아직 알려지지 않거나 어느 정도까지만 알려져 있는 양이나 정보에 대한 상징적인 이름이다.

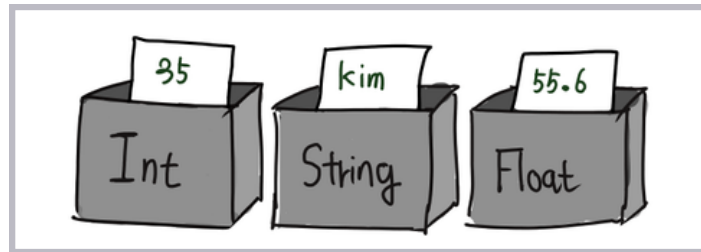
컴퓨터 소스 코드에서의 변수 이름은 일반적으로 데이터 저장 위치와 그 안의 내용물과 관련되어 있으며 이러한 것들은 프로그램 실행 도중에 변경될 수 있다.



변수

● 변수란?

하나의 값을 저장할 수 있는 메모리 번지에 붙인 이름



1. 타입

- ① 장점 : 가독성, 실수/오류 방지, 메모리 효율성 등
- ② 종류
 - 기본타입 이란 : 실제 값을 가지는 타입 이며 byte, short, int, long, float, double, char, boolean 키워드가 존재합니다.
 - 참조타입 이란 : 참조 주소값을 가지는 타입 이며 추후에 참조타입 챕터 에서 배울 예정입니다.

2. 변수명

컴퓨터가 변수저장시 메모리 번지/주소 사용하지만 개발자는 식별하기 위해 문자/이름

- ① 첫글자는 소문자인 카멜표기법 권장
 - 클래스 : SportsCar, Scanner
 - 객체/변수/배열/함수 : sportsCar, scanner
- ② 첫글자는 숫자로 시작 불가능
- ③ 특수문자는 \$ _ 만 포함 가능

3. = 대입연산자

- 오른쪽 데이터를 왼쪽에 대입 및 저장 합니다. a = 10;
- 선언과 동시에 대입시 초기화 라고 합니다. int a = 10;

4. 초기값

- 타입 범위내 데이터만 저장 가능합니다. 만약에 int 타입이면 int타입의 허용 범위내 데이터만 저장 가능합니다.

● 변수 선언 과 초기화

1. 타입 변수명 [변수 선언시 초기값[처음값] 없으면 호출 불가능]
2. 타입 변수명 = 초기값;

변수 선언 과 초기화

```
int hour = 3 ;
int minute = 5 ;
```

● 변수의 값 호출

1. 변수명 작성하여 변수내 값을 호출 합니다.

변수 값 호출

```
System.out.println( hour );
System.out.println( minute );
```

● 변수의 값 수정

1. 변수명 호출하여 새로운 값을 대입 합니다. 변수는 하나의 값만 저장 되므로 기존 데이터는 지워집니다.

변수 값 수정

```
hour = 5 ;
minute = 37;
```

● 변수 의 연산

1. 변수 호출하여 연산이 가능합니다.
2. 연산 결과를 또 다른 변수에 대입이 가능합니다.

변수 의 연산

```
int totalMinute = (hour*60) + minute ;
System.out.println( totalMinute );
```

자료형

• [정수]

- ① byte 1바이트 -128 ~ 127
- ② short 2바이트 +-3만정도
- ③ int 4바이트 +-21억정도 - 정수의기본타입 [* 직접 입력한 값(리터럴)]
- ④ long 8바이트 +-21억이상 - 정수 리터럴의 기본타입은 int 이므로 리터럴 데이터 뒤에 l/L 붙여 long 타입을 알립니다.

byte타입 [-128 ~ 127] : 1바이트 => 8bit -> 1[부호]+7[값] -> 2의7승

```
byte b1 = -128; System.out.println("b1 : " + b1);
byte b2 = 127; System.out.println("b2 : " + b2);
// byte b3 = 200; System.out.println("b3 : " + b3); // 허용범위 벗어남.
// java: incompatible types: possible lossy conversion from int to byte
```

short타입 [-32768 ~ 32767] : 2바이트

```
short s1 = 32000; System.out.println("s1 : " + s1);
// short s2 = -50000; System.out.println("s2 : " + s2); // 허용범위 벗어남.
// java: incompatible types: possible lossy conversion from int to short
```

*int타입 [+-21억정도] : 4바이트 : !!! : 정수타입 *리터럴 (그 값 자체)

```
int i1 = 10; System.out.println("i1 : " + i1);
//int i2 = 30000000000; System.out.println("i2 : " + i2); // 허용범위 벗어남.
// java: integer number too large
```

long타입 [+-21억이상] : 8바이트 : !!! : 정수타입 리터럴 뒤에 l/L

```
long l1 = 30000000000L; System.out.println("l1 : " + l1);
```

• [문자/정수]

- ① char 2바이트 0 ~ 65535, 유니코드, ' ' 사용, unsigned사용[부호를 안쓰고 양수만 사용하므로 정수부분을 더 크게 사용합니다.]

• [문자열/참조타입]

- ① String 문자열길이만큼의 크기 자동할당, 기본타입이 아닌 참조타입/클래스, " " 사용, jdk13이상에서 "" "" "" 표현식 사용가능

char [0~65535] : 2바이트 : ' ' 작은따옴표, 65536개 문자 표현가능, 부호[signed]/부호없음[unsigned]

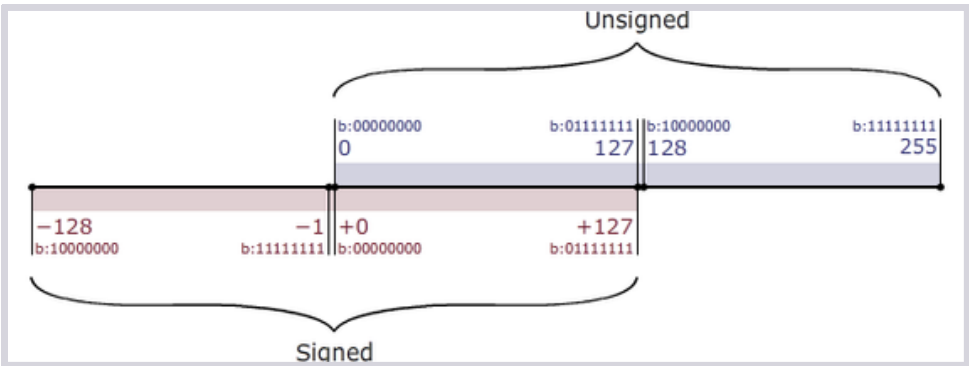
```
char c1 = 'A'; System.out.println("c1 : " + c1);
char c2 = '가'; System.out.println("c2 : " + c2);
//char c3 = "가"; System.out.println("c3 : " + c3); // 오류발생
int i3 = 'A'; System.out.println("i3 : " + i3); // 65
int i4 = '가'; System.out.println("i4 : " + i4); // 44032
char c4 = 80; System.out.println("c4 : " + c4); // P
```

String : " " 큰따옴표, 참조타입/클래스

```
String str1 = "안녕하세요"; System.out.println("str1 : " + str1);

// java/jdk 13 이후 가능한 문법
String str4 = ""
나는 자바를
학습합니다.
나는 자바 고수가 될 겁니다.
""; System.out.println("str4 : " + str4);
```

+ signed 와 unsigned



자료형

● [논리]

- ① boolean 1바이트 true 혹은 false만 저장 합니다.
 - true or false 만 표현 하므로 1bit 사용 가능하지만. 보통 컴퓨터는 한번의 읽고/쓰기 할때 1바이트 단위 이므로 boolean 도 1바이트 사용합니다.
 - 하지만 오라클 JDK 공식문서에서는 실제로 boolean의 크기를 정확하게 정의하지 않습니다.
 - 자바는 boolean 연산을 지원하지 않기 때문에 크기의 중요하게 생각하지 않습니다.

boolean [true 또는 false] : 1바이트

```
boolean bool1 = true;      System.out.println("bool1 : " + bool1);
```

● [실수]

- ① float 4바이트 7자리 유효 [* 리터럴 데이터 뒤에 f/F 붙여 float 타입을 알림]
- ② double 8바이트 15자리 유효 [* 직접 입력한 값 -> 리터럴] - 실수의기본타입]

float[소수점 8자리 표현] : 4바이트 : 반올림.

```
float f1 = 0.123456789123456789f;      System.out.println("f1 : " + f1);
```

double[소수점 17자리 표현] : 8바이트 : 실수타입의 기본타입

```
double d1 = 0.123456789123456789123456789;      System.out.println("d1 : " + d1);
```

● 자바에서 사용하는 리터럴(입력한 데이터 그 자체)

- 정수(int), 실수(double), 논리(boolean), 문자(""), 문자열(" ")

● 기본자료형으로 표현이 불가능한 범위 데이터는 문자열타입(String클래스) 활용 합니다.

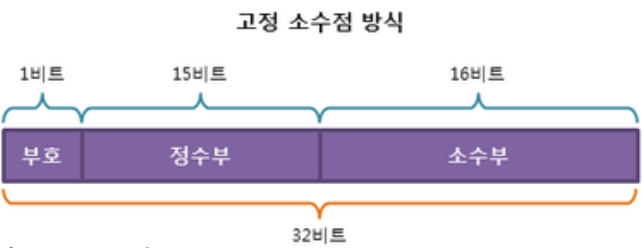
문제점 : 컴퓨터는 "정확히" 실수 표현이 불가능해 근사 과정에서 연산과정 중 오차가 쌓임에 따라 최종적으로 가시적인 오차가 발생

```
System.out.println( 0.1 + 0.2);           //  0.30000000000000004

float float1 = 123456789123456789123456789f; // 1.2345679E26
```

● 컴퓨터가 소수점 실수(real number) 표현하는 방법

- ① 고정 소수점 : 소수점이 찍힐 위치를 미리 정해 놓고 소수를 표현하는 방식 (정수 + 소수)
장점 : 실수를 정수부와 소수부로 표현하여 단순하다.
단점 : 표현의 범위가 너무 적어서 활용하기 힘들다. (정수부는 15bit, 소수부는 16bit)



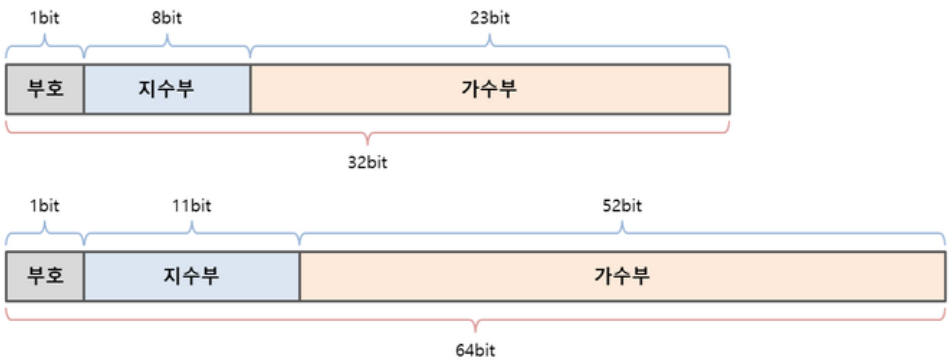
- ② 부동 소수점 : 소수점이 움직이는 방식을 활용한 실수 표현 방식, 즉 소수점의 위치가 고정되어 있지 않는다.
자바는 IEEE 754 표준 사용합니다.

- float : [부호부 1Bit | 지수부 8Bit | 가수부 23Bit] = 총 32Bit
- double : [부호부 1Bit | 지수부 11Bit | 가수부 52Bit] = 총 64Bit

타입	부호	지수부	가수부	총 비트수
Half precision	1	5	10	16
Single precision	1	8	23	32
Double precision	1	11	52	64
x86 extended precision	1	15	64	80
Quad precision	1	15	112	128

$$N = (-1)^S \times M \times 2^E$$

부호부 (Sign) : 1비트. 숫자의 부호를 나타내며, 양수일 때 0, 음수일 때 1이 됩니다.
지수부 (Exponent) : 8비트. 지수를 나타냅니다. 소수 자릿수 위치
가수부 (Mantissa) : 23비트. 가수 또는 유효숫자를 나타냅니다.



● 오차를 줄이는 방법 , 라이브러리를 사용하자. 즉 BigDecimal 클래스를 이용하면 된다

- BigDecimal value = new BigDecimal("0.1");



타입변환

● 자동 타입 변환 : 작은 타입 허용 범위가 큰 타입 허용 범위 내 대입 될 때

- ① 변환 방향 : byte → short → int → long → float → double

자동 타입 변환 : byte→short→int→long→float→double

```
byte b1 = 10;           // -128 ~ 127 까지 저장 가능 한 변수에 10 저장
int i1 = b1;            // byte변수를 int변수에 대입 [ 자동 ]
char c1 = '가';         // 0~65000 까지 저장 가능 한 변수에 '가' 저장
i1 = c1;                // char변수를 int변수에 대입 [ 자동 ]
int i2 = 50; long l1 = i2; // int → long [ 자동 ]
long l2=100; float f1 = l2; // long → float [ 자동 ]
float f2=10.5f; double d1 = f2; // float → double [ 자동 ]
```

● 강제 타입 변환 · 캐스팅 : 큰 타입 허용 범위가 작은 타입 허용 범위 내 대입 될 때

- ① 변환 방향 : byte ← short ← int ← long ← float ← double
② 큰 타입 에서 작은 타입으로 변환 하므로 데이터 손실이 있을 수 있습니다. 목적은 원래 값이 유지되면서 타입만 변환 입니다.

강제 타입 변환 : byte←short←int←long←float←double

```
int i3 = 10; byte b2 = (byte)i3; // (byte) int → byte [ 강제 ]
long l3 = 300; int i4 = (int)l3; // (int) long → int [ 강제 ]
int i5 = 65; char c2 = (char)i5; // (char) int → char [ 강제 ]
double d2 = 3.14; int i6 = (int)d2; // (int) double → int [ 강제 ]
```

+ 문자열을 기본타입으로 변화하는 함수들

변환 타입	사용 예
String → byte	String str = "10"; byte value = Byte.parseByte(str);
String → short	String str = "10"; short value = Short.parseShort(str);
String → int	String str = "10"; int value = Integer.parseInt(str);

String → long	String str = "10"; long value = Long.parseLong(str);
String → float	String str = "10"; float value = Float.parseFloat(str);
String → double	String str = "10"; double value = Double.parseDouble(str);
String → boolean	String str = "10"; boolean value = Boolean.parseBoolean(str);

● 연산식에서 자동 타입변환

- ① 피연산자 중 큰 타입을 결과타입 반환 , 단 byte 와 short는 무조건 int 로 변환 됩니다.

연산식에서 자동 타입변환

```
byte result1 = 10+20;

byte v1 = 10; byte v2 = 20;
int result2 = v1 + v2; // byte + byte => int

byte v3 = 10; int v4 = 100; long v5 = 1000L;
long result3 = v3 + v4 + v5; // byte + int => int + long => long

char v6 = 'A'; char v7 = 1;
int result4 = v6 + v7; // char + char => int

int v8 = 10;
int result5 = v8 / 10; // int / int => int

int v9 = 10;
double result6 = v9 / 4.0; // int / double => double

int v10 = 1;
int v11 = 2;
//int result7 = v10 / v11; // 0 int / int => int [ 문제 있음 - 결과에 소수점 표현이 불가능 하다. ]
double result7 = (double) v10 / v11; // 0.5 int / int => (캐스팅) 해서 소수점 표현 하자.
```

● 문자열 타입을 기본 타입 으로 변환 , 기본타입을 문자열 타입 으로 변환

- ① 문자열 타입을 기본 타입 으로 변환 , 기본타입클래스명.parse~~~(문자열)
② 기본타입을 문자열 타입 으로 변환 , String.valueOf() 또는 + ""

문자열 타입을 기본 타입 으로 변환 , 기본타입을 문자열 타입 으로 변환

```
int value1 = Integer.parseInt( "10" );
double value2 = Double.parseDouble("3.14");
boolean value3 = Boolean.parseBoolean("true");

String str1 = String.valueOf( 10 );
String str2 = 10+"";
```



입출력

● 콘솔 출력

- ① System.out.print()
- ② System.out.println()
- ③ System.out.printf("형식문자열" , 변수1/리터럴1 , 변수2/리터럴2)

출력

```
System.out.print(" 출력문구1 ");
System.out.print(" 출력문구2 ");
```

출력후 줄바꿈

```
System.out.println(" 출력문구3 ");
System.out.println(" 출력문구4 ");
```

형식문자열에 맞추어 값 출력

```
int value = 123;
System.out.printf( "상품의 가격 : %d원 \n" , value ); // %d 위치에 value 변수 출력
System.out.printf( "상품의 가격 : %6d원 \n" , value ); // %6d : 6칸 정수 자리에 value 변수 출력 [ 자릿수가 비어 있으면 공백처리 ]
System.out.printf( "상품의 가격 : %-6d원 \n" , value ); // %6d : 6칸 정수 자리에 value 변수 출력 [ 자릿수가 비어 있으면 공백처리 ]
System.out.printf( "상품의 가격 : %06d원 \n" , value ); // %6d : 6칸 정수 자리에 value 변수 출력 [ 자릿수가 비어 있으면 0 채움 ]
System.out.printf( "반지름 파이 %f \n" , 3.14 );
System.out.printf( "반지름 파이 %.1f \n" , 3.14 );
System.out.printf( "회원 아이디 : %s \n" , "qweqwe" );
```

+ 인텔리제이 에서 제공하는 출력 관련 자동완성

```
// 1. sout
System.out.println();
// 2. souf
System.out.printf("");
// 3. soutm : 현재 함수명 출력
System.out.println("Step2.main");
// 4. soutp : 현재 함수의 매개변수 출력
System.out.println("args = " + Arrays.toString(args));
// 5. soutv : 가장 가까운 변수의 값 출력
System.out.println("job = " + job);
```

● 콘솔 입력

- Scanner scanner = new Scanner(System.in); 입력객체 : { } 마다 1번 선언 하고 키보드로부터 입력받은 바이트를 저장
- 입력객체를 이용한 입력받은 값 호출할때 마다 객체내 저장된 바이트를 각 메소드에 맞춰 형변환해서 호출

- ① scanner.nextBoolean()
- ② scanner.next()
- ③ scanner.nextByte()
- ④ scanner.nextShort()
- ⑤ scanner.nextInt()
- ⑥ scanner.nextLong()
- ⑦ scanner.nextFloat()
- ⑧ scanner.nextDouble()

입력

```
Scanner scanner = new Scanner( System.in );
String 문자열 = scanner.next();
System.out.println("입력된 문자열은 : " + 문자열 );
boolean 논리 = scanner.nextBoolean();   System.out.println("입력된 논리 : " + 논리 );
byte 바이트 = scanner.nextByte();   System.out.println("입력된 바이트 : "+ 바이트 );
short 쇼트 = scanner.nextShort();   System.out.println("입력된 쇼트 : " + 쇼트);
int 인트 = scanner.nextInt();   System.out.println("입력된 인트 : " + 인트 );
long 롱 = scanner.nextLong();   System.out.println("입력된 롱 : " + 롱);
float 플롯 = scanner.nextFloat();   System.out.println("입력된 플롯 : " + 플롯 );
double 더블 = scanner.nextDouble();   System.out.println("입력된 더블 : " + 더블);
```

● 이스케이프/제어 문자 : 문자 출력을 제어하는 문자

이스케이프/제어 문자

```
System.out.println(" \ " );
System.out.println(" \ ' "); // ' : 문자
System.out.println(" \ " );
System.out.println(" \t ");
System.out.println(" \n줄바꿈");
System.out.println(" \r캐리지 리턴");
```



+더보기

- 이스케이프/제어 문자 : 문자 출력을 제어하는 문자

이스케이프 시퀀스	기능
<code>\n</code>	다음 줄로 넘어간다.
<code>\t</code>	탭을 삽입한다.
<code>\r</code>	캐리지 리턴을 삽입한다. <small>*캐리지 리턴(carriage return): 현재 줄의 처음으로 되돌아간다.</small>
<code>\f</code>	폼 피드를 삽입한다. <small>*폼 피드(form feed): 프린트 출력에 사용되며 커서를 다음 페이지의 시작 부분으로 넘긴다.</small>
<code>\b</code>	백스페이스 문자를 삽입한다. <small>*컴파일러에 따라 문자가 삭제되거나 커서가 한 칸 뒤로 이동한다.</small>
<code>\'</code>	작은따옴표를 출력한다.
<code>\"</code>	큰따옴표를 출력한다.
<code>\\</code>	역슬래시 기호(\)를 출력한다.

- 형식 지정자 : `printf()` 에서 사용 됩니다.

– %[플래그][너비][정밀도]변환문자

[플래그]	플래그	기능
	-	왼쪽으로 정렬한다. '-' 플래그가 없으면 오른쪽으로 정렬한다.
	+	양수 값에 + 기호를 추가한다. 숫자 값에만 적용된다.
	#	8진수 변환 문자와 함께 쓰면 출력 앞에 0이 붙고, 16진수 변환 문자와 함께 쓰면 출력 앞에 0x가 붙는다.
	0	앞의 빈자리에 0을 채워 넣는다. 숫자 값에만 적용된다.
	(음수에 괄호를 추가한다. 숫자 값에만 적용된다.
	,	숫자에 지역별 구분 기호를 추가한다. 숫자 값에만 적용된다.
		공백. 양수 앞에는 빈 칸 하나가 추가된다. 숫자 값에만 적용된다.

[너비] : 인수를 출력하기 위해 너비를 지정합니다. 이 너비는 출력할 최소 문자 수를 말합니다. 출력 길이가 너비 값보다 작으면 공백을 추가합니다.

[정밀도] : 부동 소수점 타입과 함께 사용되며, 정밀도를 사용해서 출력될 소수점 수를 지정할 수 있습니다.

[변환문자]	변환 문자	출력 형태
	d	부호 있는 10진 정수
	f	부호 있는 10진 실수
	o	부호 없는 8진 정수
	x, X	부호 없는 16진 정수
	e, E	과학적 표기법 기반의 실수
	g, G	값에 따라서 %e 혹은 %f를 선택함
	s	문자열
	c	문자
	b	논리형(true 또는 false)

- 진수 : 데이터 표현하는 진법 단위 – 표현 단위 다형성.

2진수 : 0 또는 1 <---> 이진코드 <---> 기계어

8진수 : 0 1 2 3 4 5 6 7

10진수 : 0 1 2 3 4 5 6 7 8 9 <---> 실생활에서 주로 사용

16진수 : 0 1 2 3 4 5 6 7 8 9 A(10) B(11) C(12) D(13) E(14) F(15)

- 용량 : 데이터 크기 정보의 단위

bit/비트 : 0 또는 1, 1비트 -> 01010101 -> 8비트(비트 8칸 뜻)

byte/바이트 : 비트8칸 , 8비트 -> 1바이트 -> 1024바이트 (*컴퓨터가 읽고/쓰기 최소단위)

kb/킬로바이트 : 1024바이트 , 1024바이트 -> 1킬로바이트 -> 1024킬로바이트

mb/메가바이트 : 1024킬로바이트 , 1024킬로바이트 -> 1메가바이트 -> 1024메가바이트

gb/기가바이트 : 1024메가바이트 , 1024메가바이트 -> 1기가바이트 -> 1024기가바이트

- 인코딩 : 정보의 형태나 형식을 변환하는 처리나 처리 방식

– 2진수를 문자로 표하는 방법

컴퓨터 : 010101010101 <--> 사람 : ??????????????

컴퓨터 : 010101010101 <--> 규칙/패턴/해독 (인코딩) <---> 사람 : A

– **아스키코드** : 7비트(확장8비트) -> 128문자 표현 1바이트 = 특수문자/숫자/영문

– **유니코드** : 전세계 공용어 문자 표현 2바이트 = 특수문자/숫자/영문/한글

