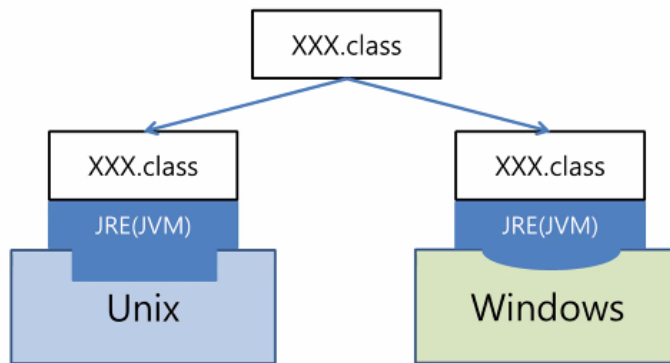


## Java 소개-특징(1/2)

### ➤□1. Platform Independent

- Class File format(JVM spec에 정의되어 있음)
- Java Virtual Machine



1. JVM(Java Virtual Machine) 을 통해 운영체제에 상관없이 사용 가능  
-> Platform Independent

### JRE(Java Runtime Environment) : 컴퓨터의 운영체제 소프트웨어 상에서 실행되고 클래스 라이브러리 및 특정 JAVA 프로그램이 실행해야 하는 기타 리소스를 제공하는 소프트웨어 계층

-> JRE는 Java 프로그램의 개발과 실행을 위한 세 개의 상호 연관된 컴포넌트 중 하나

나머지 두 개의 컴포넌트

- JDK( Java Development Kit ) : Java 애플리케이션의 개발을 위한 툴 세트

Java 프로그램의 실행이 Java 프로그램 개발 프로세스의 일부이므로, 모든 JDK에는 항상 호환 가능한 JRE가 포함

- JVM (Java Virtual Machine) : 라이브 Java 애플리케이션을 실행

JRE는 JDK를 사용하여 작성된 Java 코드를 JVM에서 이를 실행하는 데 필요한 필수 라이브러리와 결합한 후 결과 프로그램을 실행하는 JVM의 인스턴스를 작성합니다. JVM은 다수의 운영체제에 사용 가능하며, JRE를 사용하여 작성된 프로그램이 이 모두에서 실행됩니다. 이러한 방식으로, JRE(Java Runtime Environment)는 수정 없이도 어떤 운영체제에서든 Java 프로그램을 실행할 수 있도록 함.

이클립스 설정

JDK 경로 설정

Window – Preferences – Installed JREs – Add - Standard VM - 다운로드 받은 jdk 폴더 선택

Encoding 설정

Window – Preferences – encoding – Workspace – UTF – 8 ( CSS, HTML, JSP 도 변경해주자 )

프로젝트 생성 순서

File -> New -> Project -> Java Project -> 프로젝트명 작성 -> Create module-info.java file 체크  
해제

클래스 생성 순서

src -> new -> class -> **클래스명은 대문자로 시작!** -> public static void main(String[] args) 체크

**### Java 파일명과 클래스명 동일**

자바 파일을 저장하면 이클립스가 자동으로 Class 파일 생성해줌!!

Class 파일 생성 순서는

Jdk/bin -> java.exe -> javac.exe을 거쳐 class 파일 생성해준다 (내가 설정해둔 work-space/bin 폴더 확인)

## Java 소개-특징(2/2)

### ➤ 2. OOP

- 재사용성이 뛰어남
- ☐유지보수 용이

### ➤ 3. Simple & Easy

- 클래스 상속은 단일상속만 지원
- ☐포인터 지원하지 않음(직접적으로 메모리 접근을 못하게 함)

### ➤ 4. Garbage Collector

- ☐Heap 메모리 영역의 객체들을 정리해줌

### ➤ 5. Database Independent(JDBC)

- ☐JDBC interface를 이용하면 DB에 독립적

### ➤ 6. Multi-Threading

OOP (Object Oriented Programming) : 객체 지향 프로그래밍

운영체제에서 메모리 와 프로세스 관리를 함 -> Java 에서는 Garbage Collector을 통하여 할당된 메모리를 반납한다

JDBC interface 를 통해 프로젝트 내 DB 변동이 있어도 DB 연동에 정보 수정만 해준다면 다른 수정 없이 사용 가능

## Java 특징-플랫폼소개(1/2)

---

### ➤ J2SE(Java 2 Platform Standard Edition)

- ☐ 일반적인 범용 컴퓨터 환경을 지원하는 플랫폼
- ☐ JDBC ,RMI ,Applet , Application, ...

### ➤ J2EE(Java 2 Platform Enterprise Edition)

- ☐ Enterprise 환경을 지원하는 플랫폼
- ☐ Servlet/JSP ,EJB ,JavaMail, JTA ,JNDI, ...

### ➤ J2ME(Java 2 Platform Micro Edition)

- 일반적인 컴퓨팅 환경에 비해 자원 제약이 많은 핸드폰, PDA 등의 소형 기기 환경을 지원하는 플랫폼
- ☐ MIDlet, Java Card, ...

## Chapter 2. 변수(Variable)

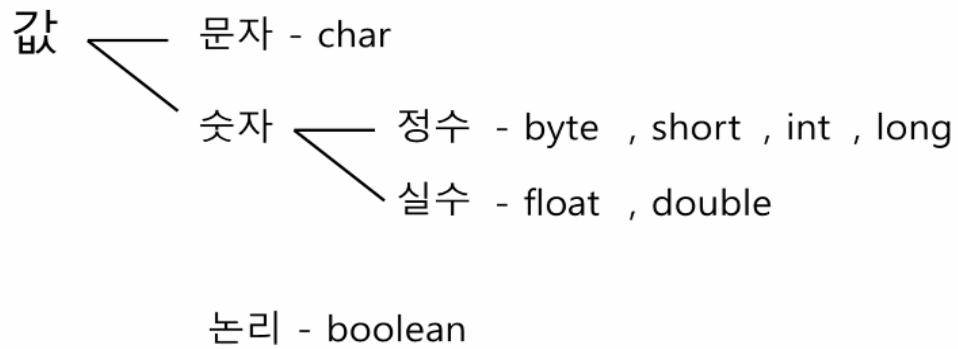
---

1. 변수(Variable)란?
2. 변수의 타입(Type)
3. 변수의 선언방법
4. 명명규칙(Naming Convention)
5. 변수, 상수, 리터럴
6. 리터럴과 접미사
7. 변수의 기본값과 초기화
8. 문자와 문자열
9. 정수의 오버플로우(Overflow)
10. 형변환(Casting)

## Chapter 2. 변수(Variable)

---

### 2. 변수의 타입(Data type)



실수 - double 이 float 보다 크기에 오차가 적다

## 4. 명명규칙(Naming convention)

- ▶ 대소문자가 구분되며 길이에 제한이 없다.
  - True와 true는 서로 다른 것으로 간주된다.
- ▶ 예약어(Reserved word)를 사용해서는 안 된다.
  - true는 예약어라 사용할 수 없지만, True는 가능하다.
- ▶ 숫자로 시작해서는 안 된다.
  - top10은 허용하지만, 7up은 허용되지 않는다.
- ▶ 특수문자는 '\_'와 '\$'만을 허용한다.
  - \$sharp은 허용되지만 S#arp는 허용되지 않는다.

변수는 알파벳 대소문자, 숫자, 특수문자 조합으로 이루어짐

단 숫자는 앞으로 올 수 없고 특수문자는 '\_' , '\$' 만 사용 가능

## 4. 명명규칙 - 권장사항

- ▶ 클래스 이름의 첫 글자는 항상 대문자로 한다.
  - 변수와 메서드 이름의 첫 글자는 항상 소문자로 한다.
- ▶ 여러 단어 이름은 단어의 첫 글자를 대문자로 한다.
  - lastIndexOf, StringBuffer
- ▶ 상수의 이름은 대문자로 한다. 단어는 '\_'로 구분한다.
  - PI, MAX\_NUMBER





## 5. 변수, 상수, 리터럴

- ▶ 변수(variable) – 하나의 값을 저장하기 위한 공간
- ▶ 상수(constant) – 한 번만 값을 저장할 수 있는 공간
- ▶ 리터럴(literal) – 그 자체로 값을 의미하는 것

```
int score = 100;  
score = 200;  
char ch = 'A';  
String str = "abc";  
final int MAX = 100;  
MAX = 200; // 에러
```

Final : 상수 // 모든 데이터 타입이 가능 -> 한번 선언된 변수는 변경이 불가능 -> 상수!~

## 2. 변수의 타입(Data type)

### ▶ 기본형(Primitive type)

- 8개 (boolean, char, byte, short, int, long, float, double)
- 실제 값을 저장

### ▶ 참조형(Reference type)

- 기본형을 제외한 나머지(String, System 등)
- 객체의 주소를 저장(4 byte, 0x00000000~0xffffffff)

참조형은 값이 저장된 객체의 주소를 저장 (= 메모리의 주소를 저장?)

## Chapter 2. 변수(Variable)

### 기본형(Primitive type)

- ▶ 논리형 - true와 false중 하나를 값으로 갖으며, 조건식과 논리적 계산에 사용된다.
- ▶ 문자형 - 문자를 저장하는데 사용되며, 변수 당 하나의 문자만을 저장할 수 있다.
- ▶ 정수형 - 정수 값을 저장하는데 사용된다. 주로 사용하는 것은 int와 long이며, byte는 이진데이터를 다루는데 사용되며, short은 c언어와의 호환을 위해 추가되었다.
- ▶ 실수형 - 실수 값을 저장하는데 사용된다. float와 double이 있다.

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

↑

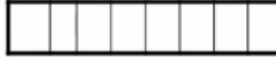
나이는 기본형 변수 byte를 사용하여 저장해도 되지만 최적화를 위해 기본형변수 int 에 저장

## Chapter 2. 변수(Variable)

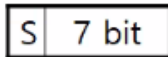
1 bit



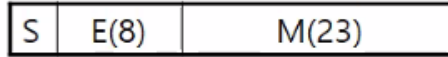
8 bit = 1 byte



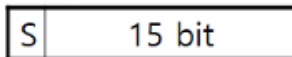
byte  $-2^7 \sim 2^7-1$



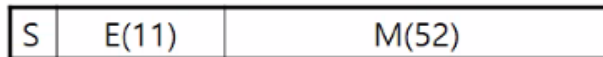
float  $1+8+23=32$  bit = 4 byte



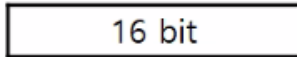
short  $-2^{15} \sim 2^{15}-1$



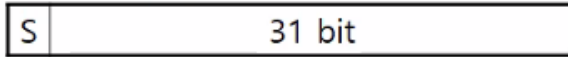
double  $1+11+52=64$  bit = 8 byte



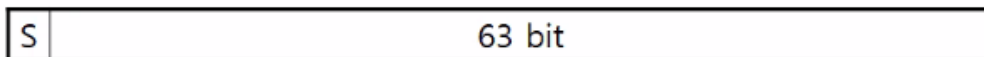
char  $0 \sim 2^{16}-1$



int  $-2^{31} \sim 2^{31}-1$



long  $-2^{63} \sim 2^{63}-1$



## Chapter 2. 변수(Variable)

1. byte  $\rightarrow$  int

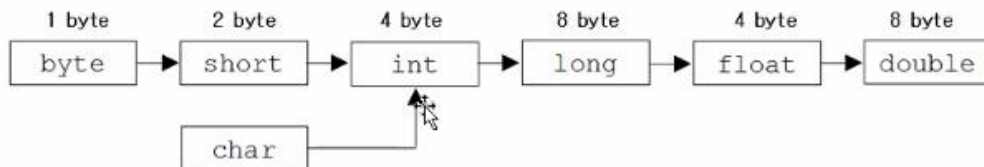
```
byte b = 10;
```

```
int i = (int)b // 생략가능
```

2. int  $\rightarrow$  byte

```
int i2 = 300;
```

```
byte b2 = (byte)i2; // 생략불가
```

[illegible]

자동 형 변환 순서

반대는 수동으로 캐스팅 해줘야 함

### Chapter 3. 연산자(Operator)

---

- |                            |                  |
|----------------------------|------------------|
| 1. 연산자(Operator)란?         | 11. 비트연산자(&, ,^) |
| 2. 연산자의 종류                 | 12. 논리연산자(&&,  ) |
| 3. 연산자의 우선순위               | 13. 삼항연산자(? :)   |
| 4. 증감연산자(++/--)            | 14. 대입연산자(=,op=) |
| 5. 부호연산자(+,-)와 논리부정연산자(!)  |                  |
| 6. 비트전환연산자(~)              |                  |
| 7. 이항연산자의 특징               |                  |
| 8. 나머지 연산자(%)              |                  |
| 9. 쉬프트연산자(<<,>>,>>>)       |                  |
| 10. 비교연산자(>,<,>=,<=,==,!=) |                  |

## 1. 연산자(Operator)란?

- ▶ 연산자(Operator)
  - 어떠한 기능을 수행하는 기호(+, -, \*, / 등)
- ▶ 피연산자(Operand)
  - 연산자의 작업 대상(변수, 상수, 리터럴, 수식)

$$a + b$$

피 연산자가 2개 이상 필요로 하는 연산자를 이항 연산자


피 연산자가 1개 필요로 하는 연산자를 단항 연산자

## Chapter 3. 연산자(Operator)

---

### 2. 연산자의 종류

▶ 단항 연산자 : + - (타입) ++ -- ~ !

▶ 이항 연산자  산술 : + - \* / % << >> >>>  
비교 : > < >= <= == !=  
논리 : && || & ^ |

▶ 삼항 연산자 : ? :

▶ 대입 연산자 : =



종 류	연산방향	연 산 자	우선순위
단항 연산자	←	++ -- + - ~ ! (타입)	높음
산술 연산자	→	* / %	
	→	+ -	
	→	<< >> >>>	
비교 연산자	→	< > <= >= instanceof	
	→	== !=	
논리 연산자	→	&	
	→	^	
	→		
	→	&&	
	→		
삼항 연산자	→	?:	
대입 연산자	←	= *= /= %= += -= <<= >>= >>>= &= ^=  =	낮음

### 3. 연산자의 우선순위<sub>(3/4)</sub>

- ▶ 상식적으로 생각하라. 우리는 이미 다 알고 있다.

ex1)  $-x + 3$           단항 > 이항

ex2)  $x + 3 * y$       곱셈, 나눗셈 > 덧셈, 뺄셈

ex3)  $x + 3 > y - 2$       산술 > 비교

ex4)  $x > 3 \ \&\& \ x < 5$       비교 > 논리

ex5) `int result = x + y * 3;`      항상 대입은 맨 끝에

## 4. 연산자의 우선순위<sub>(4/4)</sub>

▶ 그러나 몇 가지 주의해야 할 것이 있다.

1) <<, >>, >>>는 덧셈연산자보다 우선순위가 낮다.

ex5)  $x \ll 2 + 1$                        $x \ll (2 + 1)$  과 같다.

2) ||, |(OR)는 &&, &(AND)보다 우선순위가 낮다.

ex6)  $x < -1 \parallel x > 3 \ \&\& \ x < 5$

$x < -1 \parallel (x > 3 \ \&\& \ x < 5)$  와 같다.

## 7. 이항연산자의 특징<sup>(1/7)</sup>

이항연산자는 연산을 수행하기 전에 피연산자의 타입을 일치시킨다.

- ▶ int보다 크기가 작은 타입은 int로 변환한다.

( byte, char, short → int )

- ▶ 피연산자 중 표현범위가 큰 타입으로 형변환 한다.

byte + short → int + int → int

char + int → int + int → int

float + int → float + float → float

long + float → float + float → float

float + double → double + double → double

단축키

syso + Ctrl + Space -> System.out.println();

Ctrl + Shift + F -> 자동 줄 정리

Ctrl + F11 -> 실행