**Java [Object Oriented Languages]**

자바(영어: Java, 문화어: 쟈바)는 썬 마이크로시스템즈의 제임스 고슬링(James Gosling)과 다른 연구원들이 개발한 객체 지향적 프로그래밍 언어이다. 처음에는 가전제품 내에 탑재해 동작하는 프로그램을 위해 개발했지만 현재 웹 애플리케이션 개발에 가장 많이 사용하는 언어 가운데 하나이고, 모바일 기기용 소프트웨어 개발에도 널리 사용하고 있다.

자바의 문법적인 특성은 C 언어와 비슷하다. 자바가 다른 컴파일 언어와 구분되는 가장 큰 특징은 컴파일 된 코드가 플랫폼 독립적이라는 것이다. 자바 컴파일러는 자바 언어로 작성된 프로그램을 바이트코드라는 특수한 바이너리 형태로 변환한다. 바이트코드를 실행하기 위해서는 JVM(자바 가상 머신, Java Virtual Machine)이라는 특수한 가상 머신이 필요한데, 이 가상 머신은 자바 바이트코드를 어느 플랫폼에서나 동일한 형태로 실행시킨다. 때문에 자바로 개발된 프로그램은 CPU나 운영 체제의 종류에 관계없이 JVM을 설치할 수 있는 시스템에서는 어디서나 실행할 수 있으며, 이 점이 웹 애플리케이션의 특성과 맞아떨어져 폭발적인 인기를 끌게 되었다.

출처 : Wikipedia(<https://ko.wikipedia.org/wiki/>자바\_(프로그래밍\_언어))

1. **Data type**

Java에서 지원하는 data type은 기본 타입(primitive type)과 참조 타입(reference type)으로 분류할 수 있다.

1. **기본 타입(primitive type)**
   1. **정수형**

* 정수형 data type의 종류에는 byte, short, int, long 4가지가 있다.

|  |  |
| --- | --- |
| **Data type** | **Data value size** |
| byte | 1byte |
| short | 2byte |
| int | 4byte |
| long | 8byte |

예를 들어 설명하기 위해, X는 10, Y는 20이라고 한다.

* Arithmetic Operators

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Operator 양쪽의 값을 더한다. | X+Y의 값은 30 |
| - | 왼쪽 피연산자의 값에서 오른쪽 피연산자의 값을 뺀다. | X-Y의 값은 -10 |
| \* | Operator 양쪽의 값을 곱한다. | X\*Y의 값은 200 |
| / | 왼쪽 피연산자의 값을 오른쪽 피연산자의 값으로 나눈다. | Y/X의 값은 2 |
| % | 왼쪽 피연산자의 값을 오른쪽 피연산자의 값으로 나눈 나머지 값을 취한다. | Y%X의 값은 0 |
| ++ | 피연산자의 값을 1증가시킨다. | X++의 값은 11 |
| -- | 피연산자의 값을 1감소시킨다. | X--의 값은 9 |

* Relational Operators

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| == | 두 피연산자의 값을 비교하고 같으면 true이다. | (A == B) 는not true. |
| != | 두 피연산자의 값을 비교하고 아니면 true이다. | (A != B) 는true. |
| > | 왼쪽 피연산자의 값이 오른쪽 피연산자의 값보다 큰지 확인하고 그렇다면 true이다. | (A > B) 는not true. |
| < | 왼쪽 피연산자의 값이 오른쪽 피연산자의 값보다 작은지 확인하고 그렇다면 true이다. | (A < B) 는true. |
| >= | 왼쪽 피연산자의 값이 오른쪽 피연산자의 값보다 크거나 같은지 확인하고 그렇다면 true이다. | (A >= B) 는 not true. |
| <= | 왼쪽 피연산자의 값이 오른쪽 피연산자의 값보다 작거나 같은지 확인하고 그렇다면 true이다. | (A <= B) 는 true. |

* Bitwise Operators

A는 0011 1100, B는 0000 1101이라고 가정한다.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| & | 피연산자 양쪽에 모두 1이 있을 때만 1로 바뀌고 나머지의 경우 전부 0이다. | (A & B) 의 값은 0000 1100 |
| | | 피연산자 양쪽 중 하나라도 1이 있으면 1로 바귀고 양쪽 다 0이면 0이다. | (A | B) 의 값은 0011 1101 |
| ^ | 피연산자 중 한쪽만 1인 경우에만 1이고 나머지의 경우 전부 0이다. | (A ^ B) 의 값은 0011 0001 |
| ~ | 비트 값을 반전시킨다. | (~A ) 의 값은 1100 0011 |
| << | 왼쪽 피연산자 값은 오른쪽 피연산자가 지정한 비트 수만큼 왼쪽으로 이동합니다. | A << 2 의 값은1111 0000 |
| >> | 왼쪽 피연산자 값은 오른쪽 피연산자가 지정한 비트 수만큼 오른쪽으로 이동합니다. | A >> 2 의 값은 1111 |
| >>> | 왼쪽 피연산자 값은 오른쪽 피연산자에 의해 지정된 비트 수만큼 오른쪽으로 이동되고 시프트 값은 0으로 채워집니다. | A >>>2 의 값은 0000 1111 |

* Logical Operators

A의 값이 true라고 하고 B의 값은 false라고 가정한다.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| && | 양쪽 피연산자가 전부 0이 아니면 true이다. | (A && B) 는 false |
| || | 두 피연산자 중 하나만 0이 아니어도 true이다. | (A || B) 는 true |
| ! | 해당 피연산자의 logical state를 반전시킨다. | !(A && B) 는 true |

* Assignment Operators

A는 2이고, B는 3이라고 가정한다.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | 오른쪽 피연산자의 값을 왼쪽 피연산자에 대입한다. | C = A + B 는 2+3의 값인 5를 C에 대입한다. |
| += | 왼쪽 피연산자의 값에 오른쪽 피연산자의 값을 더한 값을 왼쪽 피연산자에 대입한다. | C += A 는 C+A의 값이 C에 들어간다. |
| -= | 왼쪽 피연산자의 값에 오른쪽 피연산자의 값을 뺀 값을 왼쪽 피연산자에 대입한다. | C -= A는 C-A의 값이 C에 들어간다. |
| \*= | 왼쪽 피연산자의 값에 오른쪽 피연산자의 값을 곱한 값을 왼쪽 피연산자에 대입한다. | C \*= A는 C\*A의 값이 C에 들어간다. |
| /= | 왼쪽 피연산자의 값에 오른쪽 피연산자의 값을 나눈 값을 왼쪽 피연산자에 대입한다. | C /= A 는 C/A의 값이 C에 들어간다. |
| %= | 왼쪽 피연산자의 값에 오른쪽 피연산자의 값을 모듈러한 값을 왼쪽 피연산자에 대입한다. | C %= A 는 C%A의 값이 C에 들어간다. |
| <<= | 왼쪽 피연산자의 값에 왼쪽 피연산자의 값을 오른쪽 피연산자의 값만큼 <<한 값을 대입한다. | C <<= 2는 C<<2의 값이 C에 들어간다. |
| >>= | 왼쪽 피연산자의 값에 왼쪽 피연산자의 값을 오른쪽 피연산자의 값만큼 >>한 값을 대입한다. | C >>= 2는 C>>2의 값이 C에 들어간다. |
| &= | 왼쪽 피연산자의 값에 왼쪽 피연산자의 값을 오른쪽 피연산자의 값만큼 &한 값을 대입한다. | C &= 2는 C&2의 값이 C에 들어간다. |
| ^= | 왼쪽 피연산자의 값에 왼쪽 피연산자의 값을 오른쪽 피연산자의 값만큼 ^한 값을 대입한다. | C ^= 2 는 C^2의 값이 C에 들어간다. |
| |= | 왼쪽 피연산자의 값에 왼쪽 피연산자의 값을 오른쪽 피연산자의 값만큼 |한 값을 대입한다. | C |= 2 는 C|2의 값이 C에 들어간다. |

* Misc Operators

Conditional Operator (?:)

variable x = (expression) ? value if true : value if false

example)

x = (x>2) ? 12:23;

x>2가 참이면 12가 x에 대입되고

x>2가 거짓이면 23이 x에 대입된다.

* 1. **실수형**
* 실수형 data type의 종류에는 float, double 2가지가 있다.

|  |  |
| --- | --- |
| **Data type** | **Data value size** |
| float | 4byte |
| double | 8byte |

**Operator**

**정수형과 같으나 비트 레벨에서의 연산은 허용되지 않는다. (&,|,^,<< 등등)**

* 1. **문자형**
* 문자형 data type에는 char가 있다.

|  |  |
| --- | --- |
| **Data type** | **Data value size** |
| char | 2byte |

* C언어와 다르게 2byte이고 유니코드 문자를 표현한다.

Operator

정수형과 같다.

* 1. **논리형**
* 논리형 data type에는 boolean이 있다.

|  |  |
| --- | --- |
| **Data type** | **Data value size** |
| boolean | true or false |

Operator

정수형의 Relational operator, Bitwise operator, Logical operator와 같다.

1. **참조 타입(reference type)**

**2.1) Array**

**Type[] name 의 형태로 사용한다.**

**2.2) enum**

**2.3) class**

**2.4) interface**

1. **Sequence control**
2. **Function**