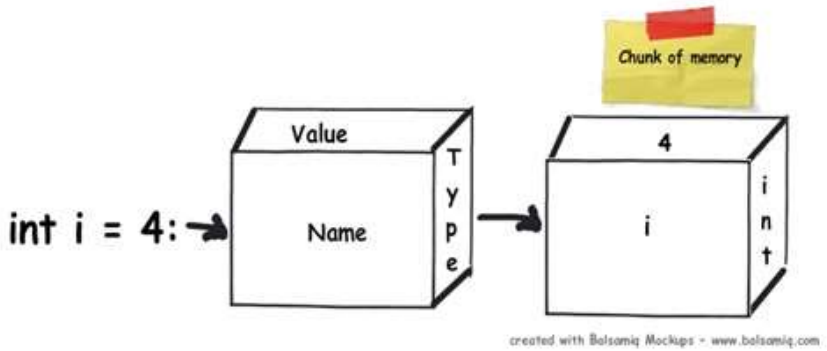
Variable

변수

Variable

변수 : 데이터를 저장하기 위해 프로그램에 의해 이름을 할당받은 메모리 공간



Variable - Type

Type : 데이터를 식별하는 분류

데이터를 저장하는 방식에 따른 분류 : 기본형, 참조형
데이터를 표현하는 방식에 따른 분류 : 논리형, 숫자형, 문자형

기본형(Primitive Type)

논리형 : boolean (1byte) * 메모리 주소의 크기가 1byte이기 때문

숫자형 : 정수형 : byte(1byte), short(2byte), int(4byte), long(8byte)
실수형 : float(4byte), double(8byte)

문자형 : char(2byte)

* 표준 정수형 타입 : int
* 표준 실수형 타입 : double

참조형(Reference Type)

사용자가 정의한 모든 타입 ex) String

Variable - Type

타입 별 데이터 범위

* ₩u0000 : 유니코드 0번
* 정수형 타입은 음수/양수를 표현하기 위해 첫 bit를 부호비트로 사용

분류	자료형	크기(byte)	기본값	값의 범위
논리형	boolean	1	false	true, false
문자형	char	2	'₩u0000'	'₩u0000' ~ '₩uffff'
정수형	byte	1	0	$-2^7 \sim 2^7 - 1$
	short	2	0	$-2^{15} \sim 2^{15} - 1$
	int	4	0	$-2^{31} \sim 2^{31} - 1$
	long	8	0L	$-2^{63} \sim 2^{63} - 1$
실수형	float	4	0.0f	$1.4 \times 10^{-45} \sim 3.4 \times 10^{38}$
	double	8	0.0d	$4.9 \times 10^{-324} \sim 1.8 \times 10^{308}$

Variable – 선언 및 할당

변수의 선언 및 할당

1. 변수 선언	int i;
2. 변수에 값을 할당	i = 10;
3. 변수의 선언과 동시에 할당	int j = 11;
4. 여러 개의 변수를 한번에 선언	int k,l,m;
5. 여러 개의 변수를 동시에 선언 및 할당	int k = 12, l = 13, m = 14;

* 4,5번은 같은 타입의 변수만 가능

Variable – 선언 및 할당

* CPU가 메모리에 저장된 데이터를 읽기 위해서는 메모리의 시작 주소와 읽을 크기(칸 수)를 알아야 한다.

0	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**
1	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**1**
1	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**0**

변수의 선언 및 할당 과정

변수에는 데이터가 저장되어 있는 메모리의 주소가 저장된다.
변수의 타입을 통해 데이터의 크기와 형태를 지정한다.

변수 : 변수의 메모리 주소를 저장
Type : 변수의 크기와 형태를 지정

```
char ch;                //메모리에 char타입의 크기만큼의 공간을 확보한 뒤 그 주소값을 ch에 반환
ch = 'A';               //ch에 저장된 메모리주소부터 char타입의 크기만큼의 공간에 'A'의 유니코드값을 저장
System.out.println(ch); //ch에 저장된 메모리 주소부터 char타입의 크기만큼 데이터를 읽어서 데이터를 반환

int num = ch;           //ch에 저장된 메모리주소부터 char타입의 크기만큼 데이터를 읽어서 int타입 변수에 저장
System.out.println(num);//ch에 저장되어있던 이진데이터를 int타입의 변수로 출력하기 때문에 65가 출력
```

Variable – 형변환(Casting)

컴퓨터의 값 처리 규칙

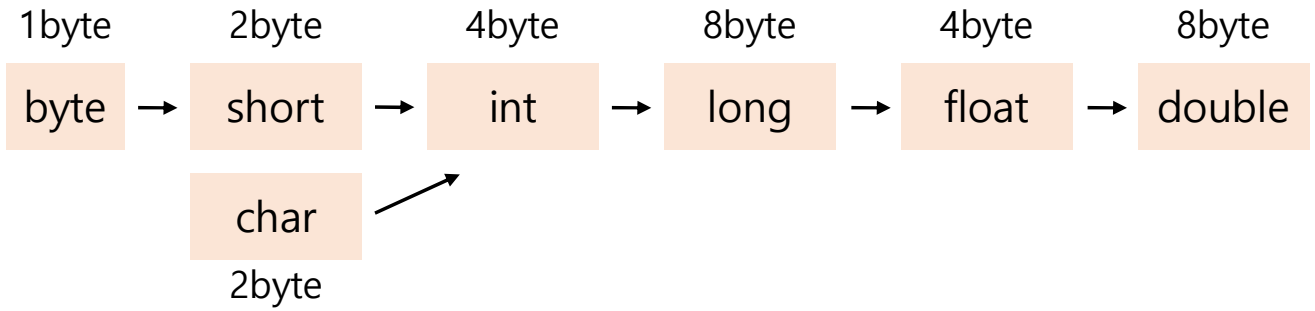
1. 같은 종류의 타입 간에만 대입 가능
2. 같은 종류의 타입 간에만 연산 가능

다른 타입 간 대입이나 연산이 필요한 경우 형변환이 필요

Variable – 형변환(Casting)

자동 형변환

컴파일러가 자동으로 크기가 작은 자료형을 크기가 큰 자료형으로 변환




```
double dnum = 10; //컴파일러가 자동으로 10.0으로 형변환한 다음 대입
System.out.println(dnum); //10.0

dnum = dnum + 11; //컴파일러가 11을 자동으로 11.0으로 형변환한 다음 연산
System.out.println(dnum); //21.0
```

Variable – 형변환(Casting)

명시적 형변환

자동 형변환이 지원되지 않는 상황에서 명시적으로 타입을 변경하는 것.

형변환 연산자를 사용해 명시적으로 형변환을 할 수 있다.

형변환 연산자 : ( )

데이터손실이 발생할 수 있다.

```
int inum = (int) 10.1;
System.out.println(inum); //10
```

Variable – Data Overflow

Data Overflow : 타입에 지정된 크기보다 큰 데이터를 변수에 할당 했을 때 발생하는 데이터 손실

byte number = 127;

0	1	1	1	1	1	1	1

number = number + 1;

0	1	1	1	1	1	1	1

1	0	0	0	0	0	0	0

byte로 표현 가능한 음수 중에 가장 작은 수 : -128

```
System.out.println(number);     // -128
```

Variable - 변수명명규칙

1. 대소문자가 구분되며 길이 제한이 없다.

2. 예약어를 사용할 수 없다.

3. 숫자로 시작할 수 없다.

ex) age1은 가능하지만 1age는 불가능

4. 특수문자는 _와 $만 허용한다.

$ 내부 클래스에서 사용

_ 사용 시 컴파일 에러는 없지만 관례상 사용하지 않는 것이 좋음

5. 변수는 CamelCase를 따른다.

ex) ageOfVampire, coffeePrice

6. 상수는 UPPER_SNAKE_CASE를 따른다.

ex) COMPANY_EMAIL, MAX_SPEED

Variable – 자바 예약어(keyword)

abstract	boolean	break	byte	case
catch	char	class	continue	default
do	double	else	extends	false
final	finally	float	for	if
implements	import	instanceof	int	interface
long	native	new	package	private
protected	public	return	short	static
super	switch	synchronized	this	throw
throws	transient	true	try	void
volatile	while			