2. 자료형

2.1 변수와 자료형

변수란? 데이터를 저장하기 위해 메모리에 공간을 생성하고 이름을 부여해야함. 이 메모리 공간에 부여되는 이름이 “변수”. 메모리 주소를 직접 사용하는 것은 어렵기 때문에 이름을 붙여 사용하는 것

자료형? 데이터를 저장하기 위해 생성하는 메모리 공간을 목적에 따라 크기와 특징을 구별해 논 것

2.1.1 자료형 선언하기

자료형의 선언은 반드시 사용하기 전에 선언

자료형의 선언은 반드시 한 번만 선언

2.1.2 변수 사용하기

변수에 자료형을 지정하여 선언한 후, 값을 대입(=입력)함

```
int a; // 변수 선언과 입력 분리
a = 3;
int b = 5; // 변수 선언과 함께 입력
```

2.2. 이름짓기

2.2.1 이름을 지을 때 지켜야하는 필수 사항

- 영문 대소 문자와 한글을 사용할 수 있다 (한글은 비추천)
- 특수문자는 밑줄(_)과 달러($) 표기만 사용할 수 있다
- 아라비아 숫자를 사용할 수 있다. 단, 첫 번째 글자로는 사용할 수 없다
- 자바에서 사용하는 예약어는 사용할 수 없다

2.2.2 이름을 지을 때 지키면 좋은 권장 사항

- 영문 소문자로 시작한다
- 영문 단어를 2개 이상 결합할 때는 새로운 단어의 첫 글자를 대문자로 한다(Camel Case 낙타표기법 - userName)

- 상수는 대문자_대문자 형태 (MT_DATA_NUMBER)
- 함수(메서드)는 동사로 시작하여 기능을 설명할 수 있으면 좋음. 반드시 ( )를 이름뒤에 붙임.

  { } 중괄호로 실행코드를 감싸줌

2.2.3 변수의 생존기간 (Lifetime)

생존기간이란 변수가 만들어진 이후 사라지기까지의 기간을 의미

자바에서는 개발자가 변수를 직접 생성하고 삭제는 JVM이 해줌(= Garbage Collector)

변수는 자신이 선언된 코드를 감싸고 있는 { } 안에서만 메모리에 존재함

즉, 코드를 닫는 역할을 하는 } 표기를 만나면 메모리에서 삭제됨

2.3 자료형의 종류

2.3.1 기본자료형과 참조자료형의 차이

- 기본자료형은 소문자, 참조자료형은 대문자로 시작
- 기본자료형은 스택 메모리에 생성된 공간에 실제 값을 저장
- 참조자료형은 실제 데이터는 힙 메모리에 저장하고 스택메모리의 변수공간에는 힙의 주소를 저장

2.3.2 기본 자료형의 메모리 크키와 저장할 수 있는 값의 범위

자료형		자료크기	비고
부울대수	boolean	1 byte = 8 bit	true, false
정수	byte	1 byte = 8 bit	$-2^{7} \sim 2^{7}-1$
	short	2 byte = 16 bit	$-2^{15} \sim 2^{15}-1$
	int	4 byte = 32 bit	$-2^{31} \sim 2^{31}-1$
	long	8 byte = 64 bit	$-2^{63} \sim 2^{63}-1$
실수	float	4 byte = 32 bit	$\pm(1.40 \times 10^{-45} \sim 3.40 \times 10^{38})$
	double	8 byte = 64 bit	$\pm(4.94 \times 10^{-324} \sim 1.79 \times 10^{308})$
문자(정수)	char	2 byte = 16 bit	유니코드문자$(0 \sim 2^{16}-1)$

2.3.3 boolean

실제 할당된 1비트(0,1 / false, true)만 사용하고 나머지 7비트는 미사용

2.3.4 정수자료형 - byte, short, int, long

정수를 정수자료형에 입력(대입)할 때, byte, short는 크기 범위 내의 값을 입력시 자동으로 byte, short 로 인식함. int, long형은 기본적으로 integer로 인식하기 때문에 long형의 경우 L을 뒤에 붙여서 명시적으로 long형을 지정할 수 있음.

- L을 안붙여도 에러가 발생하지 않는 이유는 자동으로 타입변환이 일어나기 때문

2.3.5 실수자료형 - float, double

float 부호비트 1, 가수비트 23, 지수비트 8

long 부호비트 1, 가수비트 52, 지수비트 11

실수를 실수자료형에 입력할때 자동으로 double로 인식함.

그래서 float 형의 경우 반드시 뒤에 F를 붙여서 명시적으로 float 지정이 필요함

```
float f = 3.5 // 에러! double로 인식하므로 크기가 작은 타입으로의 자동
float g = 3  // 정상! 이 경우 3은 정수로 인식되고 float으로 자동 타입
```

2.3.6 문자자료형 - char

자바에서 char 타입은 반드시 단 따옴표를 ' ' 사용해야 함. " "은 String 인식

문자자료형 char는 스택에 선언되고 값으로 문자를 입력해야하지만 메모리에는 문자를 입력할 수 없음. 그래서 유니코드라는 숫자 = 문자 변환표를 사용하여 실제로는 문자에 매핑되는 숫자를 입력함

```
char a = 65;
System.out.println(a); // A
a = '\u0041';   // 16진수 표기법
System.out.println(a); // A
```

2.4 기본 자료형 간의 타입 변환

JAVA는 등호(=)를 중심으로 좌우의 타입이 동일하여야 함

숫자를 저장하는 7개(boolean 제외)의 기본자료형 사이에 타입변환 가능

```
int a = 3; // int 자료형 = int 자료형
double b = 5.8; // double 자료형 = double 자료형
float c = 3.2; //(X) float 자료형 = double 자료형
long d = 3; //(O) long 자료형 = long(int) 자료형
byte e = 5; //(O) byte 자료형 = byte 자료형
short f = 8; //(O) short 자료형 = short 자료형
```

2.4.1 자동 타입 변환과 수동 타입 변환

자료형의 크기(값의 범위) 순서
byte < short/char < int < long < float < double

<자동타입변환>

- 값의 표현 범위가 넓은 쪽으로 저장되는 경우
- int 보다 작은 자료형(byte, short)에 정수를 입력하는 경우 (단, 값의 범위 값만 허용)
- 컴파일러가 자동으로 타입변환 수행(즉, 타입변환 생략 가능)

<수동타입변환>

- 값의 표현 범위가 좁은 쪽으로 저장되는 경우
- 값의 손실이 발생할 수 있음
- 직접 표기하지 않으면 오류 발생

```
float a = 3; //float 자료형 <- int 자료형
long b = 7; //long 자료형 <- int 자료형
double c = 5.3F; //double 자료형 <- float 자료형

byte a = 3; //byte 자료형 <- int 자료형
byte b = 128;(X) //값 범위를 넘어 수동타입변환
short b = 7; //short 자료형 <- int 자료형

int a = (int)3.5; //3 값의 손실 발생
float b = (float)7.5; //7.5
byte c = (byte)128; //-12
```

2.4.2 기본 자료형 간의 연산

모든 연산은 같은 자료형끼리만 가능

다른 자료형끼리 연산을 수행하는 경우 자동타입변환이 수행되어 연산 (더 큰 타입으로 연산)

연산의 결과는 컴퓨터의 연산 최소단위인 int형 이상 (byte + short = int)

```
byte 자료형 + short 자료형 = int 자료형
byte 자료형 + int 자료형 = int 자료형
short 자료형 + long 자료형 = long 자료형
int 자료형 + float 자료형 = float 자료형
long 자료형 + float 자료형 = float 자료형
float 자료형 + double 자료형 = double 자료형

int value = 5 + 3.5 // 에러
double value = 5 + 3.5 // 8.5
int value = 5 + (int)3.5 // 8
```