변 수(Variable)

변 수(Variable)

- ‘데이터의 저장과 참조를 위해 할당된 메모리 공간’에 붙인 이름.
- 단 하나의 값을 저장할 수 있는 공간.

- 데이터 저장을 위한 메모리 공간의 할당 방법.
- 할당된 메모리 공간의 접근(저장/참조) 방법.

- 변수 선언 방법.

```
int num;          or      int num = 3;
num = 3;
```

Exam.

```
1.  class TestVariable {
2.      public static void main(String[] args) {
3.          int var1; // 변수 선언
4.          int var2;
5.          int sum;
6.
7.          var1 = 3; // 초기화(접근/저장)
8.          var2 = 7;
9.          sum = var1 + var2;
10.
11.         System.out.println(var1+"+"+var2+"="+sum); //참조
12.     }
13. }
```

변수 이름 짓기 규칙

- 규칙
  - 문자, 숫자, 한글(유니코드) 사용 가능.
  - 대소문자 구별, 길이 제한 없음.
  - 예약어 사용 금지.
  - 숫자로 시작하면 안됨.
  - 특수문자는 '_'와 '$'만을 허용.
  - 공백이 포함될 수 없다.
- 권고사항
  - 모두 대문자 피할 것.
  - 생략형 피할 것.
  - 카멜 / 파스칼 / 헝가리언 표기법 사용 권장.
    - employeeNumber : 카멜
    - employee_num : 파스칼
    - intEmployeeNum : 헝가리언

예약어(키워드)

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

자료형의 종류

자료형	데이터	메모리 크기	표현 가능 범위
boolean	참과 거짓	1 바이트	true, false
char	문자	2 바이트	모든 유니코드 문자
byte	정수	1 바이트	−128 ~ 127
short		2 바이트	−32768 ~ 32767
int		4 바이트	−2147483648 ~ 2147483647
long		8 바이트	−9223372036854775808 ~ 9223372036854775807
float	실수	4 바이트	$\pm(1.40\times10^{-45} \sim 3.40\times10^{38})$
double		8 바이트	$\pm(4.94\times10^{-324} \sim 1.79\times10^{308})$

자료형(1/4)

- 논리형 : boolean
  - true : 참, false : 거짓

```
1.  class Boolean {
2.      public static void main(String[] args) {
3.          boolean b1 = false;
4.
5.          System.out.println(b1);
6.          System.out.println(3<4);
7.      }
8.  }
```

자료형(2/4)

- 문자 자료형 : char
  - 문자 하나를 2바이트로 하는 유니코드 기반 표현.
  - 유니코드는 전 세계의 문자를 표현 가능.
  - 문자는 작은 따옴표로 표현.

```
1.  class ToChar {
2.      public static void main(String[] args) {
3.          char code = 'A';
4.          char ch = 0x41;
5.
6.          System.out.println(code);
7.          System.out.println(ch);
8.      }
9.  }
```

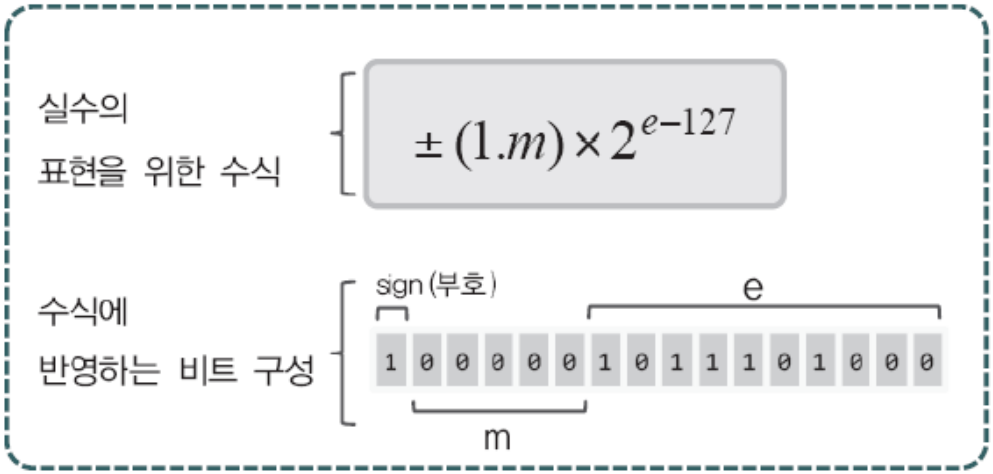
자료형(3/4)

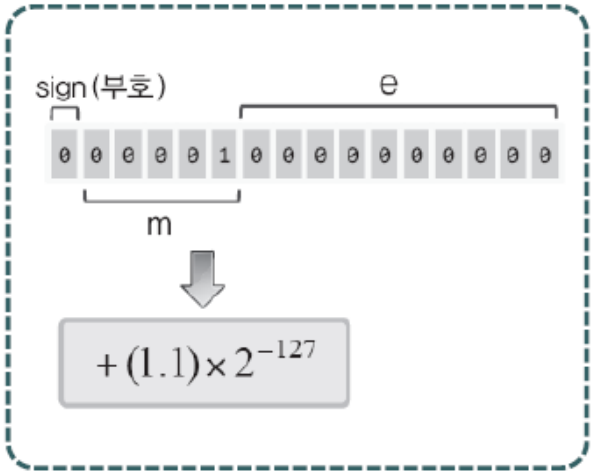
- 정수 자료형 : byte, short, int, long
  - 정수를 표현하는데 사용되는 바이트 크기에 따라서 구분됨.
  - CPU는 int형 데이터의 크기만 연산 가능.
    - short형 데이터는 int형 데이터로 자동 변환.
  - short / byte의 필요성.
    - 연산보다 데이터의 양이 중요시 되는 상황 존재.
    - mp3 파일, 동영상 파일 등등.

자료형(4/4)

- 실수 자료형 : float, double
  - float는 소수점 이하 6자리 double은 15자리 정밀도.
  - float와 double 모두 매우 충분한 표현의 범위를 자랑.
  - 이 둘의 가장 큰 차이점은 정밀도.
  - 필요한 정밀도를 바탕으로 자료형을 결정.
  - 일반적으로 double의 선택이 선호.

실수의 표현


실수의
표현을 위한 수식
$\pm(1.m)\times 2^{e-127}$
수식에
반영하는 비트 구성
sign(부호)
e
1 0 0 0 0 0 1 0 1 1 1 0 1 0 0 0
m


sign(부호)
e
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
m
$+(1.1)\times 2^{-127}$

Exam.

```
1.  class IntNDouble {
2.      public static void main(String[] args) {
3.          int num1 = 0xC1A;
4.          int num2 = 0215;
5.
6.          double e1 = 1.3e-2;
7.          double e2 = 1.3e+2;
8.
9.          System.out.println(num1);
10.         System.out.println(num2);
11.         System.out.println(e1);
12.         System.out.println(e2);
13.     }
14. }
```

상 수

상 수 (literal)

- 자료형을 기반으로 메모리 공간에 저장됨.
- 이름이 없음.
- 메모리에 저장된 상수의 값을 변경 시킬 수 없음.
- 기본적으로 모든 정수형 상수는 int형으로 표현 및 저장됨 => 연산.
- 기본적으로 모든 실수형 상수는 double형으로 표현 및 저장.

상 수 표 현

- int num = 100;
- double doubleNum = 5.5;

- int num = 10000000000; // error
- long num = 10000000000; // error
- float num = 12.45; // error

- byte num = 10;
- short num = 30;

Exam-Error(1/2)

```
1.  class SuffixConst {
2.      public static void main(String[] args) {
3.          long n1 = 10000000000;
4.          long n2 = 100;
5.
6.          double e1 = 8.625;
7.          float e2 = 8.625;
8.
9.          System.out.println(n1);
10.         System.out.println(n2);
11.         System.out.println(e1);
12.         System.out.println(e2);
13.     }
14. }
```

Exam.(2/2)

```
1.  class SuffixConst {
2.      public static void main(String[] args) {
3.          long n1 = 10000000000L;
4.          long n2 = 100;
5.
6.          double e1 = 8.625;
7.          float e2 = 8.625F;
8.
9.          System.out.println(n1);
10.         System.out.println(n2);
11.         System.out.println(e1);
12.         System.out.println(e2);
13.     }
14. }
```

자료형 변환
(Type Casting)

자료형 변환

- 의미
  - 변수 또는 리터럴의 타입을 다른 타입으로 변환하는 것.
- 방법
  - int score = (int) 85.4; // 캐스트연산자 or 형변환연산자
  - byte b = (byte) score;
- ex)

```
1.  {
2.      double  d = 100.0;
3.      int  i = 100;
4.      int result = i + (int) d;
5.
6.      System.out.println("d=" + d);
7.      System.out.println("i=" + i);
8.      System.out.println("result=" + result);
9.  }
```

```
d=100.0
i=100
result=200
```

short to int 형 변환

```
int main(String[] args)
{
    short num1=10;
    short num2=20;
    short result = num1 + num2;
    . . . .
}
```

num1(10) → 00000000 00001010
num2(20) → 00000000 00010100

int형 정수 10 → 00000000 00000000 00000000 00001010
int형 정수 20 → 00000000 00000000 00000000 00010100

int형 정수 1 → 00000000 00000000 00000000 00000001

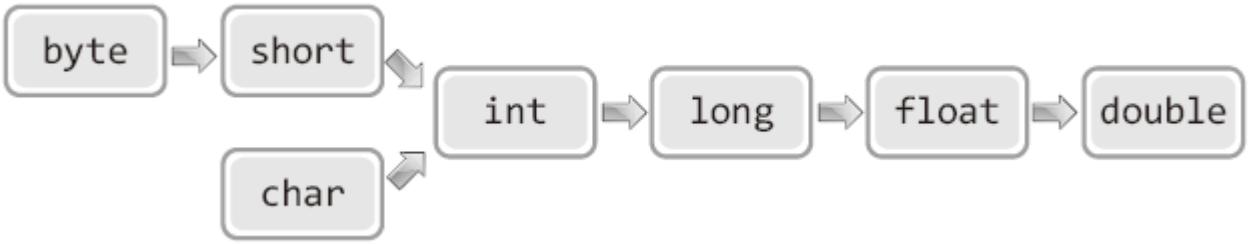
float형 실수 1.0 → 00111111 10000000 00000000 00000000

자동 형 변환(Implicit Conversion)

- short num = 10;
- double num = 10;

- int num = 0.5; // error

자동 형 변환 규칙



- float num = 10;
  // 10.0f로 자동 형 변환되어 num에 저장.

- double num = 0.5f+10;
  // 10이 10f 로 자동 형 변환 뒤 + 연산.
  // + 연산 값(float 형 10.5f)은 double형 상수 10.5로 형 변환되어
  // 변수 num에 저장.

명시적 형 변환(Explicit Conversion)

- 자동 형 변환의 규칙에 위배되지만 변환이 필요한 상황

  ex) long num1 = 6468756647L;

  int num2 = (int) num1;

- 자동 형 변환 발생 지점의 표시를 위해서

  ex) int num3 = 100;

  long num4 = (long) num3;

Exam.

```
1.  class CastingOperation {
2.      public static void main(String[] args) {
3.          char  ch1 = 'A';
4.          char  ch2 = 'Z';
5.
6.          int  num1 = ch1;
7.          int  num2 = (int) ch2;
8.
9.          System.out.println("문자 A의 유니코드 값: "+num1);
10.         System.out.println("문자 Z의 유니코드 값: "+num2);
11.     }
12. }
```