■ 컴퓨터에서 데이터 표현하기

- 0과 1로만 데이터를 저장
- bit : 컴퓨터가 표현하는 데이터의 최소 단위(2진수)이며,

  하나의 값을 저장할 수 있는 메모리의 크기
- 1byte = 8bit

10진수	2진수
0	0000000
1	0000001
2	0000010
3	0000011
4	0000100
5	0000101

10진수	16진수
9	9
10	A
11	B
12	C
13	D
14	E
15	F
16	10

```
int num = 10;
int binary = 0b1010; // 2진수 1010
int oct = 012; // 8진수 12
int hex = 0xA; // 16진수 A
```

■ 변수 (Variable)

- 값을 저장할 수 있는 메모리상의 공간
- 변수의 값은 변경될 수 있으며, 단 하나의 값만을 저장
- 객체의 상태를 나타내는 속성
- 선언 방법

```
타입  변수이름
```

```
int  age ;
```

```
double  value ;
```

■ 변수 명명규칙(Naming Convention)

작성 규칙	예
첫번째 글자는 문자이거나 '$', '_' 여야 하고 숫자로 시작할 수 없다. (필수)	가능: price, $price, _companyName 안됨: 1v, @speed, $#value
영어 대소문자가 구분된다. (필수)	firstname 과 firstName 은 다른 변수
첫문자는 영어 소문자로 시작하되, 다른 단어가 붙을 경우 첫자를 대문자로 한다. (관례)	maxSpeed, firstName, carBodyColor
문자 수(길이)의 제한은 없다.	
자바 예약어는 사용할 수 없다. (필수)	책 참조

※ 예약어

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

■ 변수 (Variable)

● 변수 값 저장

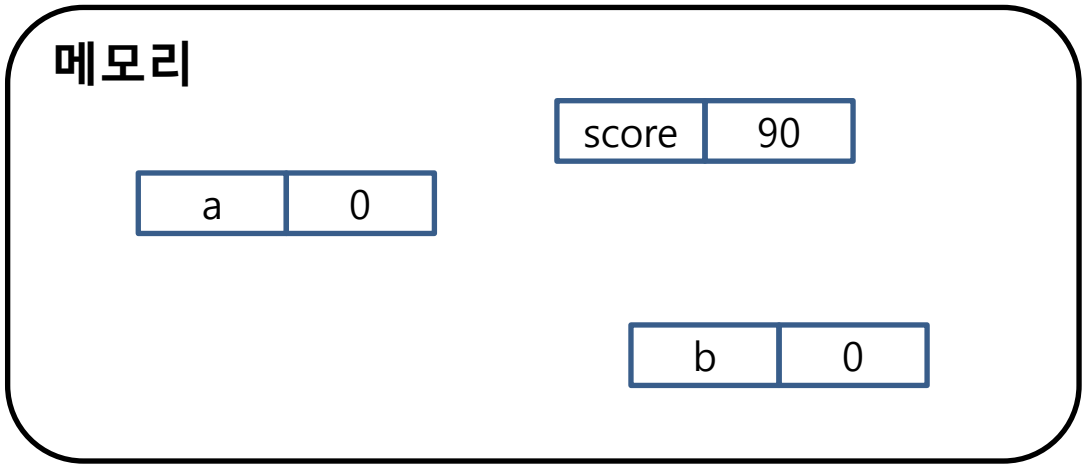
```
int score;      //변수 선언
score = 90;     //값저장
```

초기값은 변수를 선언함과 동시에 줄 수도 있다.

```
int score = 90;
```

```
int a = 0;
int b = 0;
```

int a = 0, b = 0;



■ 변수 (Variable)

● 변수 값 읽기

- 변수는 읽기 전에 반드시 초기화되어야 됨

- 잘못된 코드

```
int value;                  //변수 value 선언 (초기화 안됨)
int result = value + 10;    //변수 value 값을 읽고 10 을 더한 결과값을 변수 result 에 저장
```

- 맞게 고친 코드

```
int value = 30;             //변수 value 가 30 으로 초기화 됨
int result = value + 10;    //변수 value 값을 읽고 10 을 더한 결과값(40)을 변수 result 에 저장
```

■ 리터럴(literal)

- 소스 코드 내에서 직접 입력된 변수의 초기값

- 소스 코드 내에서 익숙해지는 것이 point !

- 종류 : 정수 / 실수 / 문자 / 문자열 / 논리

  사용 예) int a = **10**;

  float f = **3.14f;**

  boolean b = **true**;

  String s = **"문자열"**;

■ 변수 선언 및 사용 1

```
package ch02;

public class Variable1 {
    public static void main(String[] args) {
        int a = 10;
        System.out.println("a의 값은? " + a);

        int b = 11;
        System.out.println("b의 값은? " + b);

        // a의 값에 b를 입력
        // 기존 a의 값인 10은 사라짐
        a = b;
        System.out.println("a의 값은? " + a);
    }
}
```

■ 변수 선언 및 사용 2

```
public class Variable2 {
    public static void main(String[] args) {
        // 논리
        boolean isFile = false;

        // 문자
        char ch = 'A';

        // 숫자 (정수)
        byte bt = 10;
        short count = 100;
        int age = 3000;
        long ms = 20000;

        // 숫자 (실수)
        float avg = 99.99f;
        double pro = 1234.12312313d;

        // 허용 특수문자
        int abc$;
        int $abc;
        int abc_;
        int _abc;

        // 사용불가
//      int abc@;
//      int !abc;
//      int 123b;
    }
}
```

■ 기본 자료형 (Primitive Type)

- boolean (논리)
- char (문자)
- byte (정수)
- short (정수)
- int (정수)
- long (정수)
- float (실수)
- double (실수)

■ 참조 자료형 (Reference Type)

- 8개의 기본형을 제외한 나머지 타입
  - 자바 API가 제공하는 클래스
  - 사용자(개발자)가 임의로 만든 클래스
- 객체의 주소를 저장

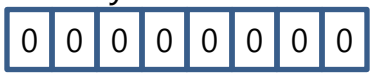
■ 기본 자료형 (Primitive Type)

● 정수, 실수, 문자, 논리 값을 저장하는 자료형

값의 종류	기본 타입	메모리 사용 크기		저장되는 값의 범위
정수	byte	1 byte	8 bit	2^7~2^7-1 (-128~127)
	char	2 byte	16 bit	0~2^{16}-1 (유니코드: ₩u0000~₩uFFFF, 0~65535)
	short	2 byte	16 bit	-2^{15}~2^{15}-1 (-32,768~32,767)
	int	4 byte	32 bit	-2^{31}~2^{31}-1 (-2,147,483,648~2,147,483,647)
	long	8 byte	64 bit	-2^{63}~2^{63}-1
실수	float	4 byte	32 bit	(+/-)1.4E-45 ~ (+/-)3.4E38
	double	8 byte	64 bit	(+/-)4.9E-324 ~ (+/-)1.7E308
논리	boolean	1 byte	8 bit	true, false

※ bit : 메모리의 최소 기억단위

1 byte = 8 bit



■ 형변환(Casting)

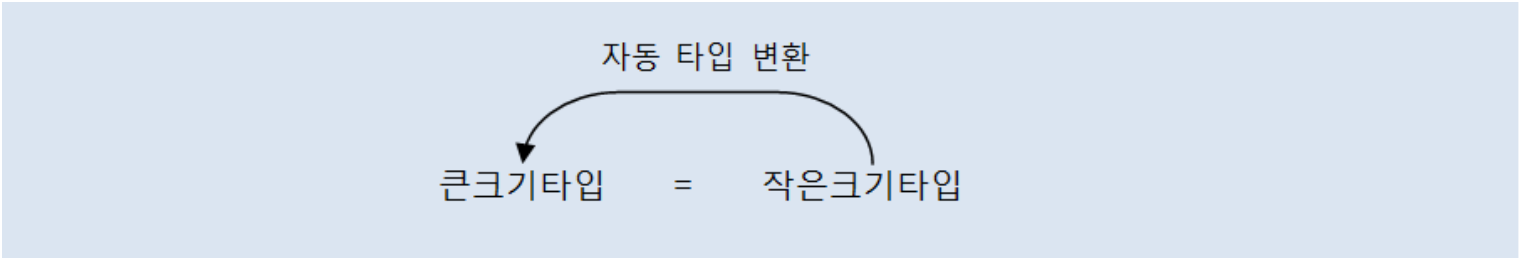
- 값의 타입을 다른 타입으로 변환
- 기본 자료형 : boolean을 제외한 나머지 7개 서로 형변환 가능
- 참조 자료형 : 같은 타입(상속) 인 경우 형변환 가능

변환	수식	결과
int → char	(char) 100	d
char → int	(int) 'd'	100
float → int	(int) 1.414	1
int → float	(float) 10	10.0f

문자	코드
…	…
0	48
1	49
…	…
A	65
B	66
…	…
a	97
b	98
…	…

■ 자동 형변환

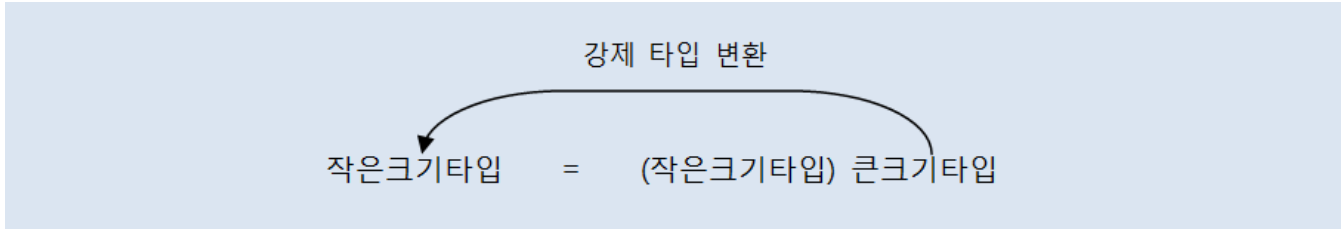
● 프로그램 실행 도중 작은 타입은 큰 타입으로 자동 타입 변환 가능



byte(1) < short(2) < int(4) < long(8) < float(4) < double(8)

■ 강제 형변환

● 큰 타입을 작은 타입으로 강제 변환



■ 연산식에서 자동 타입 변환

- 서로 다른 타입의 피연산자는 같은 타입으로 변환
- 두 피연산자 중 크기가 큰 타입으로 자동 변환

```
int intValue = 10;
double doubleValue = 5.5;

                double 타입으로 자동 변환
double result = intValue + doubleValue;       //result 에 15.5 가 저장
```

ex) int type으로 계산 결과를 얻고 싶다면?

1. double type 변수를 먼저 int로 변환 후 계산
2. 계산결과를 int로 변환

■ 형변환 예제

```
public class Casting {
    public static void main(String[] args) {
        System.out.println((int)('A')); // 65
        System.out.println((short)('a')); // 97
        System.out.println((char)(95));  // _
        System.out.println((int)(1.414)); // 1
        System.out.println((float)(10)); // 10.0

        long num1 = 100L;
        int num2 = (int) num1; // 강제 형변환
        System.out.println(num2);

        int num3 = 100;
        float num4 = 2.2f;
        // 자동 형변환
        System.out.println((num3 * num4));
        // 의도적인 표시
        double sum = 15.1 + (double)10;
        System.out.println(sum);
    }
}
```

문자	코드
…	…
0	48
1	49
…	…
A	65
B	66
…	…
a	97
b	98
…	…

■ 형변환을 사용하는 경우

- 컴파일 오류를 막기 위함

```
long num1 = 100L;
int num2 = (long) num1;
```

- 의도적인 표시

```
double sum = 15.1 + (double) 10;
```

- 형 변환 규칙에 위배되긴 하지만 변환이 필요한 상황

```
float f = 2.2F;
int i = 100;
System.out.println(  (int) ( f * i )  );
```

- 참조 자료형의 다형성 적용

■ 오버플로우(Overflow)

```
public class Overflow {
    public static void main(String[] args) {
        byte b = 127;
        b = (byte) (b + 1); // -128
        System.out.println(b);

        int num1 = 300;
        byte num2 = (byte) (num1); // 44
        System.out.println(num2);
    }
}
```