02장 변수와 타입

2.1 변수(Variable)

2.1.1 변수란?

■ 변수란 하나의 값을 저장할 수 있는 메모리 공간이다.

2.1.2 변수의 선언

```
//자료형  변수명 = 데이터;
int num = 20;
```

■ 변수명을 위한 명명 규칙(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)	firstname과 firstName은 다른 변수
첫문자는 영어 소문자로 시작하되, 다른 단어가 붙을 경우 첫자를 대문자로 한다. (관례)	maxSpeed, firstName, carBodyColor
문자 수(길의)의 제한은 없다.	
자바 예약어는 사용할 수 없다.(필수)	아래 표 참조

분류	예약어
기본 데이터 타입	boolean, byte, char, short, int, long, float, double
접근 지정자	private, protected, public
클래스와 관련된 것	class, abstract, interface, extends, implements, enum
객체와 관련된 것	new, instanceof, this, super, null
메소드와 관련된 것	void, return
제어문과 관련된 것	if, else, switch, case, default, for, do, while, break, continue
논리값	true, false
예외 처리와 관련된 것	try, catch, finally, throw, throws
기타	transient, voiatile, package, import, synchronized, native, final, static, stricip, assert

2.1.3 변수의 사용

(1) 변수값 저장

■ 변수에 값을 저장할 때에는 대입 연산자(=)를 사용한다.
■ 변수에 직접 입력된 값을 리터럴(literal)이라고 부른다.
- 정수 리터럴: 10진수(소수점이 없는 정수 리터럴), 8진수(0으로 시작되는 리터럴), 16진수(0x 또는 0X로 시작하고 0~9 숫자나 A~F 또는 a~f로 구성된 리터럴)
- 실수 리터럴: 소수점이 있는 리터럴
- 문자 리터럴: 작은 따옴표(')로 묶은 텍스트
- 문자열 리터럴: 큰 따옴표(")로 묶은 텍스트
- 논리 리터럴: true와 false

```
int score;   // 변수 선언
```

```
score = 90;  // 값 저장

int score = 90;

0, 75, -100  // 정수 리터럴(10진수)
02, -04 // 정수 리터럴(8진수)
0x5, 0xA, 0xB3, 0xAC08  // 정수 리터럴(16진수)
0.25, -3.14, 5E7, 0.17E-5  // 실수 리터럴, 5E7 = 5 x 10^7
'A', '한', '\t', '\n'  // 문자 리터럴
"대한민국", "탭 만큼 이동 \t 합니다."  // 문자열 리터럴
true, false  // 논리 리터럴
```

[LiteralExample.java]

```
package sec01.exam01_variable;

public class LiteralExample {
	public static void main(String[] args) {
		int var1 = 10;
		System.out.println(var1);

		int var2 = 010;
		System.out.println(var2);

		int var3 = 0x10;
		System.out.println(var3);
	}
}
```

(2) 변수값 읽기

■ 변수는 초기화가 되어야 읽을 수 있다.

```
int value;  // 변수 value 선언 (초기화 안 됨)
int result = value + 10;  // 컴파일 에러가 발생한다.
```

[VariableExample.java] 변수의 사용

```
public class VariableExample {
	public static void main(String[] args) {
		// 10을 변수 value의 초기값으로 저장
		int value = 10;

		// 변수 value 값을 읽고 10을 더하는 산술 연산을 수행
		// 연산의 결과값을 변수 result의 초기값으로 저장
		int result = value + 10;

		// 변수 result 값을 읽고 콘솔에 출력
		System.out.println(result);  // 20
	}
}
```

2.1.4 변수의 사용 범위

■ 변수는 선언된 블록(중괄호 {}) 내에서만 사용이 가능하다.

```
public static void main(String[] args) {
          int var1;
          if(...) {
              int var2; //var1과 var2 사용 가능
          }
          for(...) {
              int var3; //var1과 var3 사용 가능, var2는 사용 못함
          }
          //var1 사용 가능, var2와 var3는 사용 못함
}
```

[VariableScopeExample2.java] 변수의 사용 범위

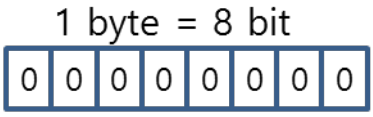
```
package sec01.exam01_variable;

public class VariableScopeExample2 {
          public static void main(String[] args) {
                    int v1 = 15;
                    if (v1 > 10) {
                              int v2;
                              v2 = v1 - 10;
                    }
                    // int v3 = v1 + v2 + 5; //v2 변수를 사용할 수 없기 때문에 컴파일 에러가 생김
          }
}
```

2.2 자료형(Data Type)

2.2.1 기본형(Primitive Type)

■ 메모리의 stack 영역에 변수명과 데이터가 저장한다. "call by value" 형태로 사용한다.



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

[꿀팁] 'Call by value'와 'Call by reference'의 차이

함수(메소드) 호출 방법은 크게 두가지가 있다.

- Call by value(값에 의한 호출)
  - 장점 : 복사하여 처리하기 때문에 안전하다. 원래의 값이 보존이 된다.
  - 단점 : 복사를 하기 때문에 메모리가 사용량이 늘어난다.

- Call by reference(참조에 의한 호출)
  - 장점 : 복사하지 않고 직접 참조를 하기에 빠르다.
  - 단점 : 직접 참조를 하기에 원래 값이 영향을 받는다.(리스크)

(1) byte 타입

- byte 타입의 메모리 사용 크기: 1 byte
- 저장되는 값의 범위: -2^7~(2^7-1)(-128~127)
- 최상위 비트(MSB: Most Significant Bit)는 정수값의 부호를 결정한다. -> 0이면 양수, 1이면 음수

(2) char 타입

- 자바는 모든 문자를 유니코드(Unicode)로 처리한다.
- 유니코드는 음수가 없기 때문에 char 타입의 변수에는 음수 값을 저장할 수 없다.
- 작은 따옴표('')로 감싼 문자를 대입하면 해당 문자의 유니코드가 저장된다.

```
char c = 'A';       //문자 A
char c = 65;        //10진수로 65
char c = '\u0041'; //16진수로 0x41, 유니코드로'\u+16진수값'표기
int uniCode = c;    //유니코드를 알고 싶다면 int형으로 변환
char c = '';        //컴파일 에러
char c = ' ';       //공백(유니코드32) 하나를 포함해서 초기화해야 한다.
```

(3) short 타입

- C언어와 호환을 위해 사용하나 비교적 자바에서는 잘 사용되지 않는다.

(4) int 타입

- 정수 연산을 위한 기본 타입이다. 예) byte형 + byte형 = int형(결과값)
- 정수값을 직접 코드에서 입력할 경우 8진수, 10진수, 16진수로 표현할 수 있다.

```
int number = 10 //10진수
int octNumber = 012; //8진수, '0'을 붙임
int hexNumber = 0xA; //16진수, '0x'를 붙임
```

(5) long 타입

■ 정수값 뒤에 소문자 'l'이나 대문자 'L'을 붙인다.

```
long var3 = 1000000000000; // 컴파일 에러
long var3 = 1000000000000L;
```

(6) float 타입

■ 부호(1bit) + 지수(8bit) + 가수(23bit) = 32bit = 4byte
예) 1.2345 = 0.12345 x 10^{1} //+(부호) M(가수) x $10^{n(지수)}$
■ 리터럴 뒤에 소문자 'f'나 대문자 'F'를 붙여야 한다.

```
float var2 = 3.14  //컴파일 에러
float var3 = 3.14F
```

(7) double 타입

■ 부호(1bit) + 지수(11bit) + 가수(52bit) = 64bit = 8byte
■ 실수 리터럴의 기본 타입이다.

```
double var1 = 3.14
```

(8) boolean 타입

■ 논리값(true/false)을 저장

2.2.2 참조형(Reference Type)

■ 객체참조형, 메모리의 heap 영역에 데이터(객체)가 저장되고, stack영역에 변수명과 주소값이 저장(참조)한다. "call by reference" 형태로 사용한다(주소값 전달에 의한 메소드 호출).
■ 종류: 클래스, 배열, 열거, 인터페이스(Collection-자료구조)
예) String s1 = "자바";
String s1 = new String("자바");

2.3 타입 변환

■ 데이터 타입을 다른 타입으로 변환하는 것
• byte ↔ int, int ↔ double
■ 종류
• 자동(묵시적) 타입 변환: Promotion (업케스팅)

- 강제(명시적) 타입 변환: Casting (다운케스팅)

2.3.1 자동 타입 변환

■ 큰 크기 타입 = 작은 크기 타입;
예) byte byteValue = 10;
int intValue = byteValue; //자동 타입 변환이 일어난다.

2.3.2 강제 타입 변환

■ 작은 크기 타입 = (작은 크기 타입) 큰 크기 타입;
예) int intValue = 103029778;
byte byteValue = (byte) intValue; //강제 타입 변환

2.3.3 연산식에서의 자동 타입 변환

■ 서로 다른 타입의 피연산자가 있을 경우, 큰 타입으로 자동 변환된 후 연산을 수행한다.
예) int intValue = 10;
double doubleValue = 5.5;
double result = intValue + doubleValue; // intValue가 double형으로 변환

■ 정수 연산일 경우, int 타입을 기본으로 한다. 특히 int보다 작은 타입은 int타입으로 변환된 후 연산이 수행된다. 따라서 연산의 결과도 int 타입이 된다.
예) byte byteValue = 10;
byte result = byteValue + byteValue //컴파일 에러가 발생
byte result = (byte)(byteValue + byteValue); //강제타입변환

[꿀팁] 자바의 자료형 변환 유형

```
1. 기본형 변환: char <--> int, int <--> double
int intValue = 97;
byte byteValue = (byte) intValue;
char charValue = (char) intValue;   // a

2. Wrapper클래스를 이용한 변환: 기본자료형 <--> 참조형
int n=10;
Integer n1 = new Integer(n)             // 박싱(boxing)
String st = n + "";                     // int -> string, boxing
int n2 = Integer.parseInt("20");        // 언박싱(unboxing)

3. 참조형 변환: 부모클래스 <--> 자식클래스
List boardlist = new ArrayList();       // 업캐스팅(up-casting), 자식클래스 -> 부모클래스
if("java".equals("jsp"){                // boolean equal(Object an), 업캐스팅
if(new Integer(30).equals(50)){         // Object an = 50; 오토박싱 + 업캐스팅
```

[과제] 확인문제

1. 자바에서 변수에 대한 설명 중 틀린 것은 무엇입니까?

(1) 변수는 하나의 값만 저장할 수 있다.
(2) 변수는 선언 시에 사용한 타입의 값만 지정할 수 있다.
(3) 변수는 변수가 선언된 중괄호({}) 안에서만 사용 가능하다.
(4) 변수는 초기값이 저장되지 않은 상태에서 읽을 수 있다.

2. 변수 이름으로 사용 가능한 것을 모두 선택하세요?

(1) modelName
(2) class
(3) 6hour
(4) $value
(5) _age
(6) int

3. 다음 표의 빈칸에 자바의 기본 타입(Primitive Type) 8개를 적어보세요.

크기/타입	1byte	2byte	4byte	8byte
정수타입	( )	( ) ( )	( )	( )
실수타입			( )	( )
논리타입	( )			

4. 다음 코드에서 타입, 변수 이름, 리터럴에 해당하는 것을 적어 보세요.

```
int age;
age = 10;
double price = 3.14;
```

타입   : ( ), ( )
변수 이름: ( ), ( )
리터럴   : ( ), ( )

5. 자동 타입 변환에 대한 내용입니다. 컴파일 에러가 발생하는 것은 무엇입니까?

```
byte byteValue = 10;
char charValue = 'A';
```

(1) int intValue = byteValue;
(2) int intValue = charValue;
(3) short shortValue = charValue;
(3) double doubleValue = byteValue;

6. 강제 타입 변환(Casting)에 대한 내용입니다. 컴파일 에러가 발생하는 것은 무엇입니까?

```
int intValue = 10;
```

```
char charValue = 'A';
double doubleValue = 5.7;
String strValue = "A";
```

(1) double var = (double) intValue;
(2) byte var = (byte) intValue;
(3) int var = (int) doubleValue;
(4) char var = (char) strValue;

7. 변수를 잘못 초기화한 것은 무엇입니까?

(1) int var1 = 10;
(2) long var2 = 10000000000L;
(3) char var3 =''; //작은 따옴표 두 개가 붙어 있음
(4) double var4 = 10;
(5) float var5 = 10;

8. 연산식에서의 타입 변환 내용입니다. 컴파일 에러가 생기는 것은 무엇입니까?

```
byte byteValue = 10;
float floatValue = 2.5F;
double doubleValue = 2.5;
```

(1) byte result = byteValue + byteValue;
(2) int result = 5 + byteValue;
(3) float result = 5 + floatValue;
(4) double result = 5 + doubleValue;