

Chapter 02 변수와 타입

2.1 변수 선언

2.2 정수 타입

2.3 문자 타입

2.4 실수 타입

2.5 논리 타입

2.6 문자열 타입

2.7 자동 타입 변환

2.8 강제 타입 변환

2.9 연산식에서 자동 타입 변환

2.10 문자열을 기본 타입으로 변환

2.11 변수 사용 범위

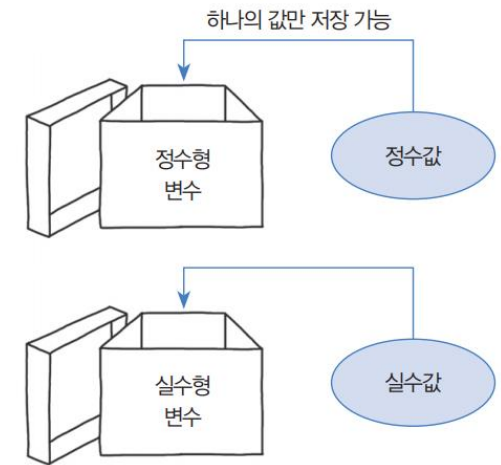
2.12 콘솔로 변수값 출력

2.13 키보드 입력 데이터를 변수에
저장

2.1 변수 선언

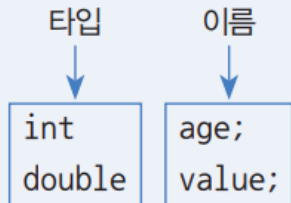
변수

- 변수(variable)란 하나의 값을 저장할 수 있는 메모리 번지에 붙여진 이름
- 자바의 변수는 다양한 타입(정수형, 실수형 등)의 값을 저장할 수 없다.

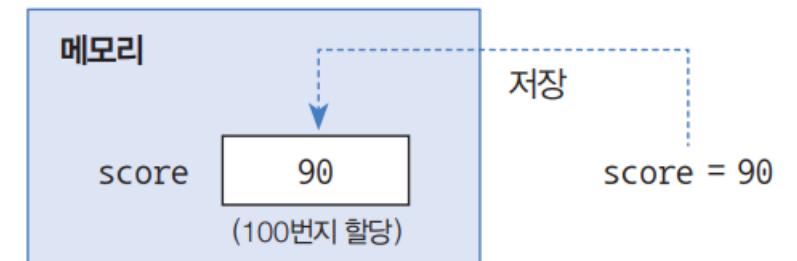


변수 선언

- 변수 변수를 사용하려면 변수 선언이 필요. 변수 선언은 어떤 타입의 데이터를 저장할 것인지 그리고 변수 이름이 무엇인지를 결정하는 것
- 변수에 최초로 값이 대입될 때 메모리에 할당 되고, 해당 메모리에 값이 저장



//정수(int) 값을 저장할 수 있는 age 변수 선언
//실수(double) 값을 저장할 수 있는 value 변수 선언



2.2 정수 타입

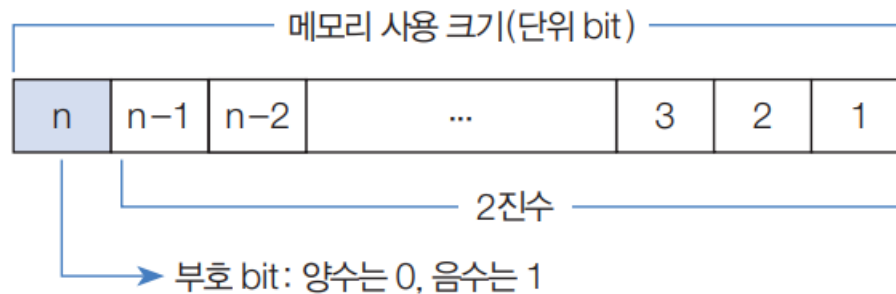
byte, short, char, int, long 타입

- 변수는 선언될 때의 타입에 따라 저장할 수 있는 값의 종류와 허용 범위가 달라짐
- 정수 타입은 5개로 메모리 할당 크기와 저장되는 값의 범위가 다름

타입	메모리 크기		저장되는 값의 허용 범위	
byte	1byte*	8bit	$-2^7 \sim (2^7-1)$	-128 ~ 127
short	2byte	16bit	$-2^{15} \sim (2^{15}-1)$	-32,768 ~ 32,767
char	2byte	16bit	$0 \sim (2^{16}-1)$	0 ~ 65535 (유니코드)
int	4byte	32bit	$-2^{31} \sim (2^{31}-1)$	-2,147,483,648 ~ 2,147,483,647
long	8byte	64bit	$-2^{63} \sim (2^{63}-1)$	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807

* 1byte = 8bit, bit는 0과 1이 저장되는 단위

- 메모리 크기를 n이라고 했을 때 정수 타입은 동일한 구조의 2진수로 저장



문자 리터럴과 char 타입

- 문자 리터럴: 하나의 문자를 작은 따옴표(')로 감싼 것
- 문자 리터럴을 유니코드로 저장할 수 있도록 char 타입 제공

```
char var1 = 'A';    //'A' 문자와 매핑되는 숫자: 65로 대입  
char var3 = '가';   //'가' 문자와 매핑되는 숫자: 44032로 대입
```

- char 타입도 정수 타입에 속함

```
char c = 65;        //10진수 65와 매핑되는 문자: 'A'  
char c = 0x0041;    //16진수 0x0041과 매핑되는 문자: 'A'
```

2.4 실수 타입

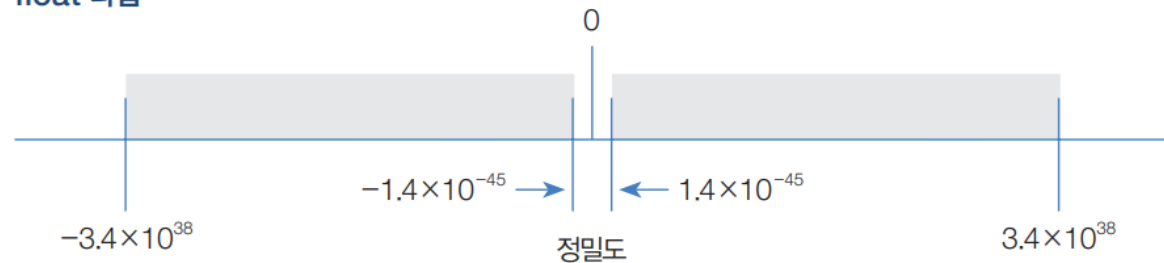
float과 double 타입

- 실수 타입에는 float과 double이 있음

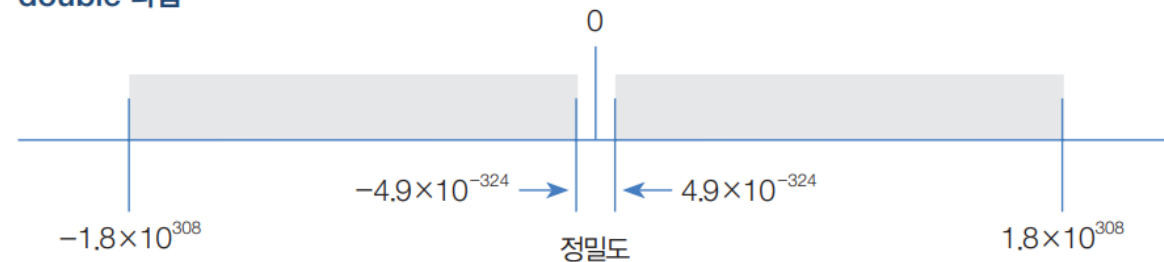
타입	메모리 크기		저장되는 값의 허용 범위(양수 기준)	유효 소수 이하 자리
float	4 byte	32 bit	$1.4 \times 10^{-45} \sim 3.4 \times 10^{38}$	7자리
double	8 byte	64 bit	$4.9 \times 10^{-324} \sim 1.8 \times 10^{308}$	15자리

- double 타입이 float 타입보다 큰 실수를 저장할 수 있고 정밀도도 높음

float 타입



double 타입



boolean 타입 변수에 대입되는 논리 타입

- 참과 거짓을 의미하는 true와 false로 구성되며 boolean 타입 변수에 대입할 수 있음

```
boolean stop = true;  
boolean stop = false;
```

- 주로 두 가지 상태값을 저장하는 경우에 사용. 조건문과 제어문의 실행 흐름을 변경하는 데 사용

	연산식	
int x = 10;		
boolean result =	(x == 20);	//변수 x의 값이 20인가?
boolean result =	(x != 20);	//변수 x의 값이 20이 아닌가?
boolean result =	(x > 20);	//변수 x의 값이 20보다 큰가?
boolean result =	(0 < x && x < 20);	//변수 x의 값이 0보다 크고, 20보다 적은가?
boolean result =	(x < 0 x > 200);	//변수 x의 값이 0보다 적거나 200보다 큰가?

문자열과 String 타입

- 문자열: 큰따옴표(“)로 감싼 문자들
- 문자열을 변수에 저장하려면 String 타입을 사용

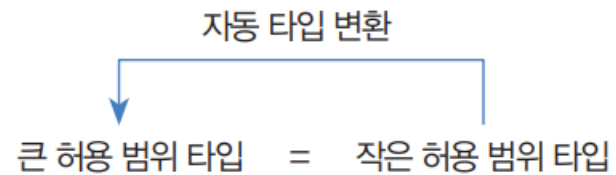
```
String var1 = "A";  
String var2 = "홍길동";
```

- 이스케이프 문자: 문자열 내부에 역슬래시(\)가 붙은 문자

이스케이프 문자	
\"	" 문자 포함
\'	' 문자 포함
\\	\ 문자 포함
\u16진수	16진수 유니코드에 해당하는 문자 포함
\t	출력 시 탭만큼 띄움
\n	출력 시 줄바꿈(라인피드)
\r	출력 시 캐리지 리턴

자동 타입 변환

- 데이터 타입을 다른 타입으로 변환하는 것
- 값의 허용 범위가 작은 타입이 허용 범위가 큰 타입으로 대입될 때 발생



`byte < short, char < int < long < float < double`

- 정수 타입이 실수 타입으로 대입되면 무조건 자동 타입 변환이 됨
- 예외: char 타입보다 허용 범위가 작은 byte 타입은 char 타입으로 자동 변환될 수 없음

캐스팅 연산자로 강제 타입 변환하기

- 큰 허용 범위 타입을 작은 허용 범위 타입으로 쪼개어서 저장하는 것
- 캐스팅 연산자로 괄호()를 사용하며, 괄호 안에 들어가는 타입은 쪼개는 단위

강제 타입 변환

작은 허용 범위 타입 = (작은 허용 범위 타입) 큰 허용 범위 타입

- 예: int → byte 강제 타입 변환

```
int intValue = 10;  
byte byteValue = (byte) intValue; //강제 타입 변환
```

```
int intValue = 10;  
byte byteValue = (byte) intValue;
```

(1byte)

00001010

10진수: 10

(4byte)

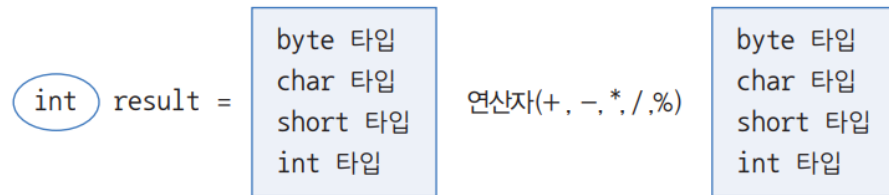
00000000 00000000 00000000 00001010

10진수: 10

원래 값이 보존됨

연산식에서 int 타입의 자동 변환

- 정수 타입 변수가 산술 연산식에서 피연산자로 사용되면 int 타입보다 작은 byte, short 타입 변수는 int 타입으로 자동 변환되어 연산 수행



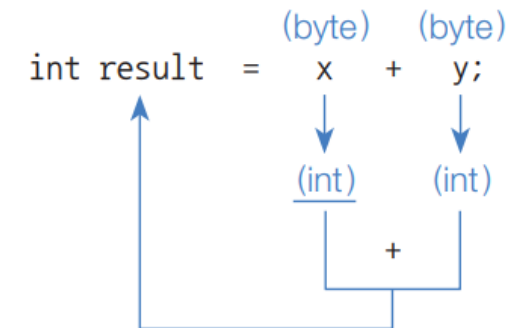
byte 타입 변수가 피연산자로 사용된 경우

```
byte x = 10;  
byte y = 20;  
byte result = x + y; //컴파일 에러  
int result = x + y;
```

int 타입 변수가 피연산자로 사용된 경우

```
int x = 10;  
int y = 20;  
int result = x + y;
```

- byte 변수가 피연산자로 사용되면 변수값은 int 값으로 연산되며, 결과값 역시 byte 변수가 아닌 int 변수에 저장해야 함



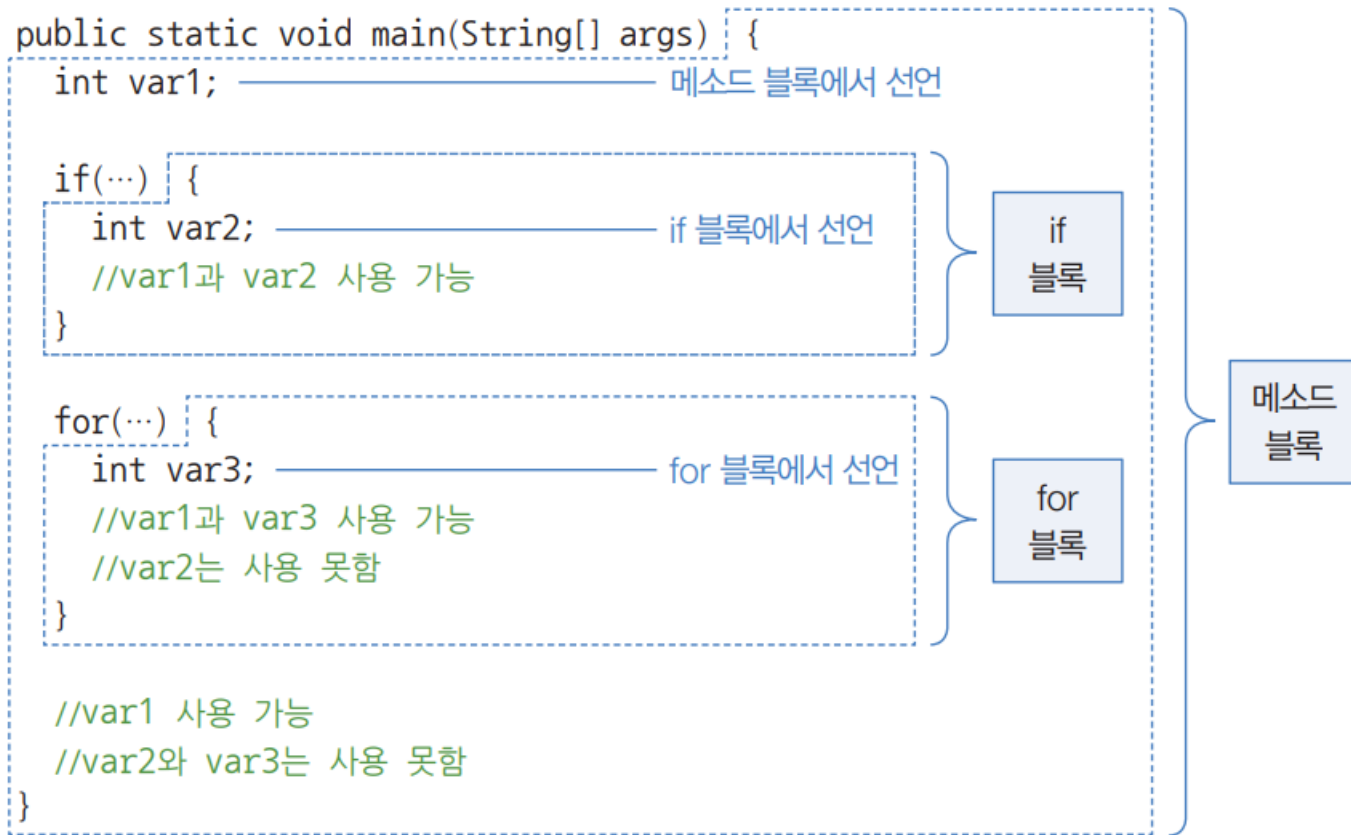
String 타입 변환하기

변환 타입	사용 예
String → byte	<pre>String str = "10"; byte value = Byte.parseByte(str);</pre>
String → short	<pre>String str = "200"; short value = Short.parseShort(str);</pre>
String → int	<pre>String str = "3000000"; int value = Integer.parseInt(str);</pre>
String → long	<pre>String str = "4000000000000"; long value = Long.parseLong(str);</pre>
String → float	<pre>String str = "12.345"; float value = Float.parseFloat(str);</pre>
String → double	<pre>String str = "12.345"; double value = Double.parseDouble(str);</pre>
String → boolean	<pre>String str = "true"; boolean value = Boolean.parseBoolean(str);</pre>

- 기본 타입의 값을 문자열로 변경할 때는 `String.valueOf()` 메소드 이용

변수 범위를 나타내는 중괄호 {} 블록


- 조건문과 반복문의 중괄호 {} 블록 내에 선언된 변수는 해당 중괄호 {} 블록 내에서만 사용 가능



Scanner 타입 변수 활용하기

- Scanner 타입 변수를 선언하고 대입 연산자 =를 사용해서 new 연산자로 생성한 Scanner 객체를 변수에 대입

생성된 Scanner를 변수에 대입




```
Scanner scanner = new Scanner(System.in);
```

scanner 변수 선언 Scanner 객체 생성

- scanner.nextLine()을 실행하면 키보드로 입력된 내용을 문자열로 읽고 좌측 String 변수에 저장

읽은 문자열을 String 변수에 저장



```
String inputData = scanner.nextLine();
```

String 변수 선언 Enter 키를 누르면 입력된 문자열을 읽음

Thank you!