

변수와 자료형

- 컴퓨터에서 데이터 표현을 학습합니다.
- 변수를 학습합니다.
- 자료형을 학습합니다.

이클립스 단축키

• 프로젝트 및 클래스 생성

- Ctrl + n + java project : 새 프로젝트 생성
- Ctrl + n + java class : 새 클래스 생성

• 프로그램 실행

- Ctrl + f11 : 프로그램 실행

• 코드 자동 정리

- Ctrl + Shift + O : 자동으로 import 하기, import 코드 정리
- Ctrl + Shift + F : 코드 자동 정리; 코드 내용을 문법 템플릿에 맞게 포매팅(들여쓰기) 해준다.

• 자동 완성

- Ctrl + Space : 자동 완성 기능
- ma+Ctrl+Space+Enter: public static void main(String[] args) {}
- sysout + Ctrl + Space : System.out.println();

• 코드 이동 및 수정

- Ctrl + D : 한줄 삭제
- Ctrl + L : 특정 소스라인으로 이동
- Alt + 방향키(위, 아래) : 특정 라인 한 줄 이동
- Ctrl + Alt + 방향키(위, 아래) : 특정 라인 코드 복사 (위 누르면 위로 복사 아래 누르면 아래로 복사)

• 주석 달기

- Ctrl + / : 한줄 주석 처리 또는 제거
- Ctrl + Shift + / : 블록 주석(/* */)
- Ctrl + Shift + W : 블록 주석 제거

• 검색

- Ctrl + H : 프로젝트 전체에서 특정 문구(텍스트) 포함 File 검색

컴퓨터에서 데이터 표현하기

- 컴퓨터는 0과 1로만 데이터를 저장 함
- bit(비트) : 컴퓨터가 표현하는 데이터의 최소 단위로 2진수 하나의 값을 저장할 수 있는 메모리의 크기
- byte(바이트) : $1\text{byte} = 8\text{bit}$

0과 1의 표현- 2진수

- 컴퓨터는 0과 1로 자료를 표현한다. 따라서 숫자나 문자도 0과 1의 조합으로 표현된다.

10진수

2진수

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

10진수와 16진수

- 2진수로 표현하면 길이가 길어지므로 8진수나 16진수를 사용하기도 한다.

10진수

16진수

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

10진수, 16진수, 8진수

- 숫자 10을 10진수, 8진수, 16진수로 출력해보자

```
1. package chapter2;
2. public class BinaryTest {
3.     public static void main(String[] args) {
4.
5.         int num = 10;
6.         int bNum = 0B1010;    // 0B : 2진수 표현
7.         int oNum = 012;       // 0 : 8진수 표현
8.         int hNum = 0XA;       // 0X : 16진수 표현
9.
10.        System.out.println(num);
11.        System.out.println(bNum);
12.        System.out.println(oNum);
13.        System.out.println(hNum);
14.    }
15. }
```

```
10
10
10
10
```

음의 정수는 어떻게 표현할까?

- 정수의 가장 왼쪽에 존재하는 비트는 부호비트입니다.
 - MSB (Most Significant Bit) 가장 중요한 비트라는 뜻
- 음수를 만드는 방법은 2의 보수를 취한다.

0	0	0	0	0	1	0	1
---	---	---	---	---	---	---	---

1의 보수를 취한다

1	1	1	1	1	0	1	0
---	---	---	---	---	---	---	---

1을 더한다

1	1	1	1	1	0	1	1
---	---	---	---	---	---	---	---

0	0	0	0	0	1	0	1
---	---	---	---	---	---	---	---

+

1	1	1	1	1	0	1	1
---	---	---	---	---	---	---	---

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

양수와 음수 더 하기

[illegible]

식별자 (identifier)

식별자 (identifier)

- **식별자란?**

- 클래스, 변수, 상수, 메소드 등에 붙이는 이름

- **식별자의 원칙**

- '@', '#', '!'와 같은 특수 문자, 공백 또는 탭은 식별자로 사용할 수 없으나 '_', '\$'는 사용 가능
- 유니코드 문자 사용 가능. 한글 사용 가능
- 자바 언어의 키워드는 식별자로 사용불가
- 식별자의 첫 번째 문자로 숫자는 사용불가
- '_' 또는 '\$'를 식별자 첫 번째 문자로 사용할 수 있으나 일반적으로 잘 사용하지 않는다.
- 불린 리터럴 (true, false)과 널 리터럴(null)은 식별자로 사용불가
- 길이 제한 없음

- **대소문자 구별**

- Test와 test는 별개의 식별자

식별자 이름 사례

• 사용 가능한 예

```
int  name;
char student_ID;           // '_' 사용 가능
void $func() { }           // '$' 사용 가능
class  Monster3 { }        // 숫자 사용 가능
int  whatsyournamemynameiskitae; // 길이 제한 없음
int  barChart; int barchart; // 대소문자 구분. barChart와 barchart는 다름
int  가격;                 // 한글 이름 사용 가능
```

• 잘못된 예

```
int  3Chapter;             // 식별자의 첫문자로 숫자 사용 불가
class  if { }              // 자바의 예약어 if 사용 불가
char false;               // false 사용 불가
void null() { }           // null 사용 불가
class  %calc { }          // '%'는 특수문자
```

자바 키워드

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

좋은 이름 붙이는 언어 관습

- 기본 : 가독성 높은 이름

- 목적을 나타내는 이름 붙이기 : s 보다 sum
- 충분히 긴 이름으로 붙이기 : AVM보다 AutoVendingMachine

- 자바 언어의 이름 붙이는 관습 : 헝가리언 이름 붙이기

- 클래스 이름

- 첫 번째 문자는 대문자로 시작
- 각 단어의 첫 번째 문자만 대문자

```
public class HelloWorld { }  
class AutoVendingMachine { }
```

- 변수, 메소드 이름

- 첫 번째 문자는 일반적으로 소문자
- 첫 단어 이후 각 단어의 첫 번째 문자는 대문자로 시작

- 상수 이름

- 모든 문자를 대문자로 표시

```
final double PI = 3.141592;
```

```
int myAge;  
boolean isSingle;  
public int getAge() { }
```

변수 (variable)

2-3. 변수

사람의 나이는 해가 바뀌면 변한다. (age)

두 수를 이용하여 사칙 연산을 하면 그 결과 값은 연산자에 따라 달라진다. (result)

게임을 하면 게임 레벨이 점점 올라간다 (level)

프로그래밍에서 값(Data)를 사용
하기 위해 선언하는 것을 **변수**라 한다.

2-3. 변수

- 프로그램에서 사용되는 자료를 저장하기 위한 공간
- 할당 받은 메모리의 주소 대신 부르는 이름
- 프로그램 실행 중에 값 변경 가능
- 사용되기 이전에 선언 되어야 한다.
- `variable` 이라 함



2-3. 변수의 선언과 사용

▶ 변수의 선언과 사용

- ▶ **변수란?** 메모리 영역에 데이터를 저장할 공간을 의미
(변수가 저장되는 공간은 JVM의 스택 영역)
- ▶ 개발자는 변수를 선언함으로써 사용할 공간 크기를 정하고
변수명과 연산자를 이용해서 데이터를 가져오거나 변경할 수 있음



- ▶ `espresso`라는 변수명에는 30이라는 값이 설정되고, 30이라는 값은 JVM 메모리에 할당됨을 의미

2-3. 변수의 선언과 초기화

```
int level ;      //level 이라는 이름의 변수 선언  
level = 10 ;
```

```
int level = 0;   //level 변수 선언과 동시에 0 으로 초기화
```

int의 역할 : level 변수의 데이터 타입을 정의
int의 의미 : level은 정수이며 4바이트의 메모리 공간을 사용한다.

2-3. 변수 선언 시 유의점

- 변수의 이름은 알파벳, 숫자, _, \$로 구성된다.
- 대소문자를 구분한다.
- 변수의 이름은 숫자로 시작할 수 없고, 키워드도 변수의 이름으로 사용할 수 없다.
- 이름 사이에 공백이 있을 수 없다.
- 변수의 이름을 정할 때는 변수의 역할에 어울리는, 의미 있는 이름을 지어야 한다.

2-3. 변수명 규칙

▶ 모두를 위한 변수명 규칙

▷ 자바에서 변수명(Variable name)을 선언할 때는 지켜야 할 공통 규칙이 있으며 이를 명명규칙(Naming convention)이라 함

▷ 반드시 지켜야 하는 명명 규칙

- 대소문자는 서로 다르게 인식하며 이름의 길이에는 제한이 없다.

예 : `int espresso = 1;`과 `int Espresso = 1;`에서 `espresso`와 `Espresso`는 서로 다른 변수명

- 미리 정해진 예약어를 변수 이름으로 사용할 수 없다.

예 : `int static = 0;`은 사용할 수 없음 `static`이라는 이름은 이미 자바 문법에서 사용되고 있는 예약어 (Abstract, Class, extends, New, while, Assert, continue, final, native, Strickfp 등)

- 변수명은 숫자로 시작할 수 없다

예 : `int 2coffee = 1;`은 사용할 수 없음, 하지만 `int coffee1 = 1;`은 사용 가능

- 특수문자는 '_'와 '\$'만 허용한다

예 : `int coffee_size = 1;`은 사용 가능하지만 `int coffee&sugar = 1;`은 불가능

2-3. 변수명 규칙 2

▶ 모두를 위한 변수명 규칙

- ▶ 다음의 규칙은 어진다고 하여도 컴파일과 애플리케이션 실행에는 아무런 문제가 없지만 코드의 통일성이나 추후 유지 보수, 타인과의 협업을 위해서 규칙을 준수하는 것을 강력히 권고

▶ 권장 명명 규칙

- 변수와 메소드 이름의 첫 글자는 소문자로 선언

예 : `int espresso = 1;`(권장), `int Espresso = 1;`(비권장)

- 여러 단어일 때 첫 번째 단어의 첫 글자는 소문자로 선언, 다음 단어의 첫 글자는 대문자로 선언

예 : `int caffeLatte = 1;`(권장), `int caffe_latte = 1;`(비권장)

- 상수일 때는 모든 글자를 대문자로 선언하고 여러 단어로 된 상수일 때는 '_' 로 구분

예 : `final int COFFEE_DEFAULT_SIZE = 1;` (권장)

- 변수명은 반드시 사용 목적을 내포

예 : `int age=33;`(권장), `int abcd = 33;` (비권장)

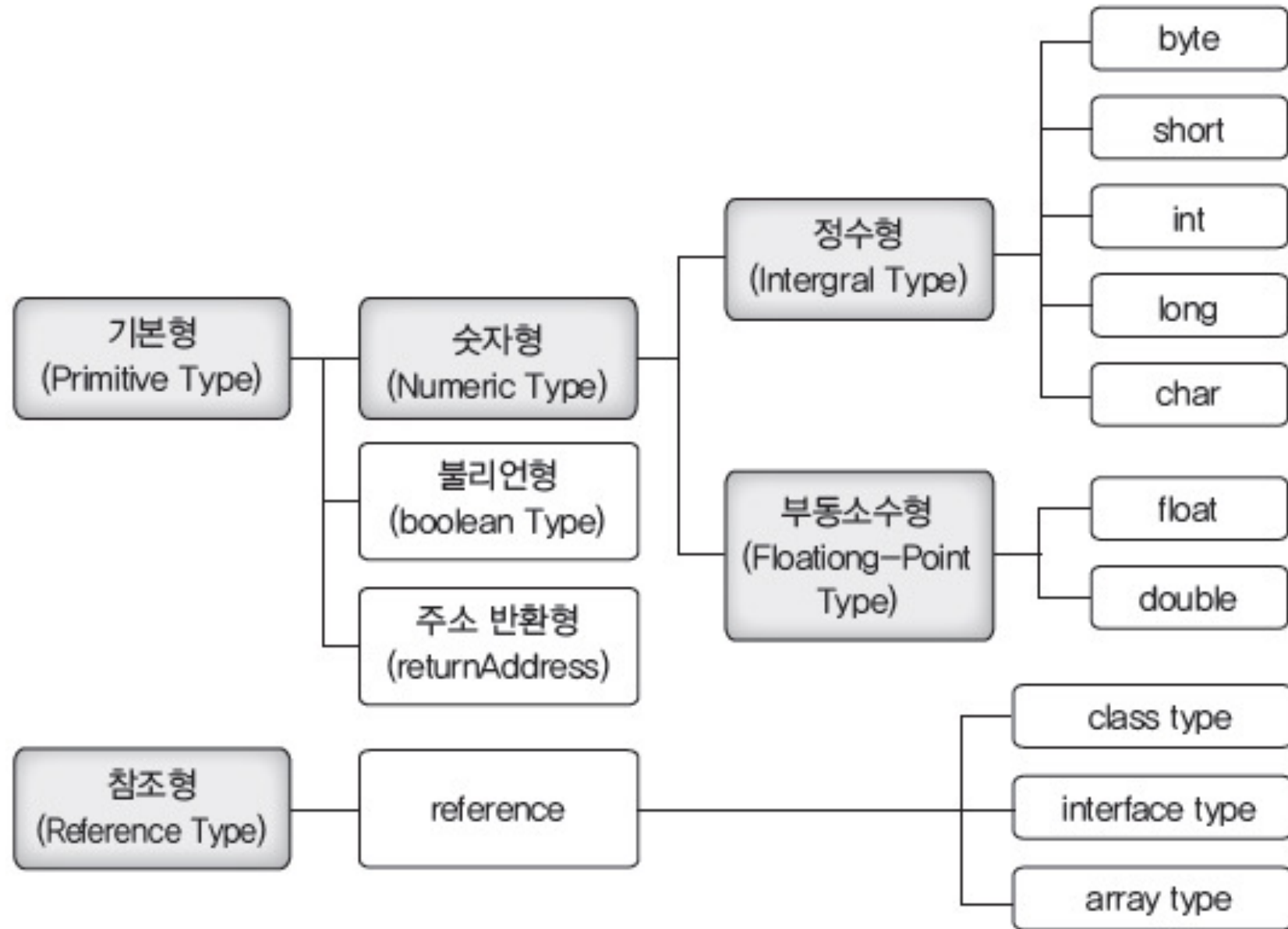
자료형 (Data Type)

자료형 (데이터형식)

▶ 데이터형의 종류와 사용법 익히기

- ▶ 자바는 엄격한 정적 타입 지정 언어이므로 반드시 데이터형(Date type)을 선언
- ▶ 데이터형은 각각 메모리 크기가 다르며 저장할 수 있는 데이터의 종류도 다름
- ▶ 그러므로 변수 데이터의 목적에 맞는 데이터형 선택이 필요
- ▶ 개발자는 자신이 사용하고자 하는 데이터에 따라서 적절한 데이터형을 선택할 수 있어야 함

자료형의 종류



자바의 데이터 타입

- **자바의 데이터 타입**

- 기본 타입 : 8 개

- boolean
 - char
 - byte
 - short
 - int
 - long
 - float
 - double

- 레퍼런스 타입 : 1 개이며 용도는 다음 3 가지

- 배열(array)에 대한 레퍼런스
 - 클래스(class)에 대한 레퍼런스
 - 인터페이스(interface)에 대한 레퍼런스

변수가 저장되는 공간의 특성 - 자료형

	정수형	문자형	실수형	논리형
1바이트	byte	-	-	boolean
2바이트	short	char	-	-
4바이트	int	-	float	-
8바이트	long	-	double	-

변수가 사용할 공간의 크기와 특성에 따라 자료형을 사용하여 변수를 선언한다.

예) `int num;`

정수 자료형

자료형	바이트 크기	수의 범위
byte	1	$-2^7 \sim 2^7 - 1$
short	2	$-2^{15} \sim 2^{15} - 1$
int	4	$-2^{31} \sim 2^{31} - 1$
long	8	$-2^{63} \sim 2^{63} - 1$

int 로 10을 표현 할 때



자료형의 크기

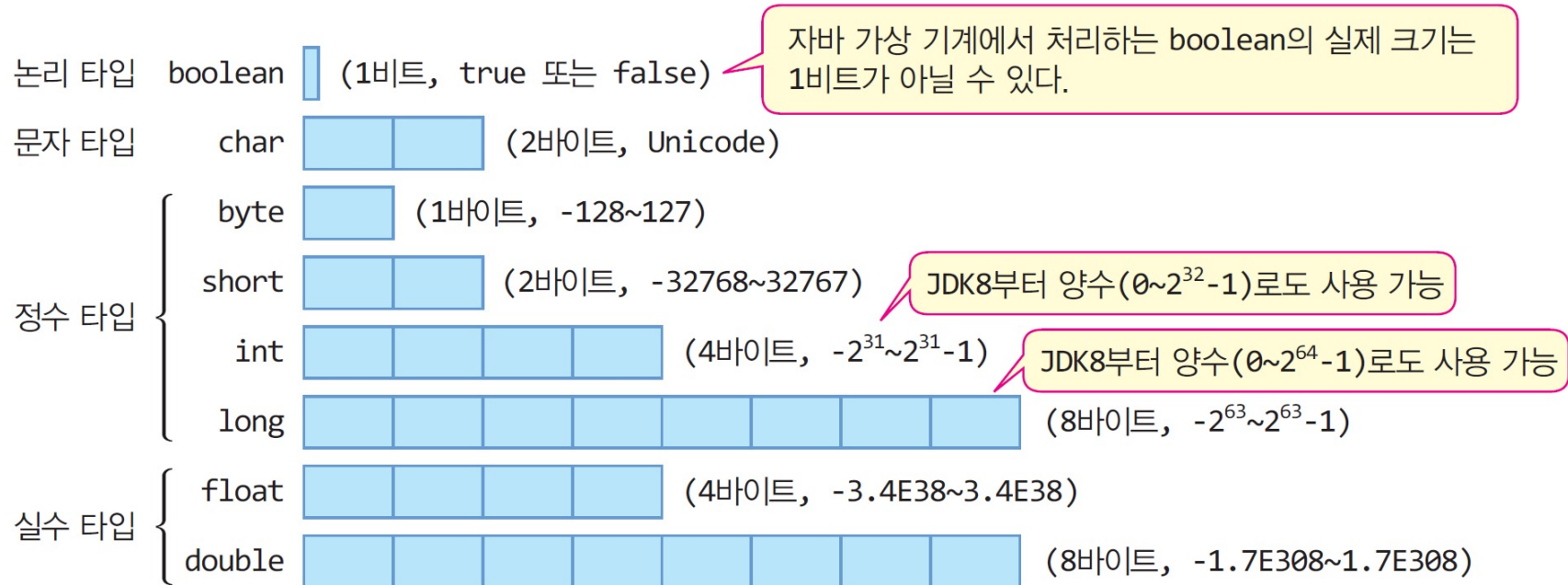
- ▷ 데이터의 크기와 성질에 따라서
반드시 변수형도 적절한 형태의 모양과 크기가 되어야 함

데이터형	메모리 크기	유효 저장 범위
byte	8bit	$-2^7 \sim 2^7 - 1$ 사이의 값
short	16bit	$-2^{15} \sim 2^{15} - 1$ 사이의 값
int	32bit	$-2^{31} \sim 2^{31} - 1$ 사이의 값
long	64bit	$-2^{63} \sim 2^{63} - 1$ 사이의 값
char	16bit	$0 \sim 2^{16} - 1$ 사이의 값
float	32bit	single-precision float
double	64bit	double-precision float

자바의 기본 타입

• 특징

- 기본 타입의 크기가 정해져 있음
 - CPU나 운영체제에 따라 변하지 않음



byte 와 short

- **byte: 1바이트 단위의 자료형**

- 동영상, 음악 파일등 실행 파일의 자료를 처리 할 때 사용하기 좋은 자료형

- **short: 2바이트 단위의 자료형**

- 주로 c/c++ 언어와의 호환 시 사용

int

- 자바에서 사용하는 정수에 대한 기본 자료 형
- 4바이트 단위의 자료 형
- 프로그램에서 사용하는 모든 숫자(리터럴) 은 기본적으로 int (4 바이트)로 저장됨
- 32 비트를 초과하는 숫자는 long 형으로 처리해야 함

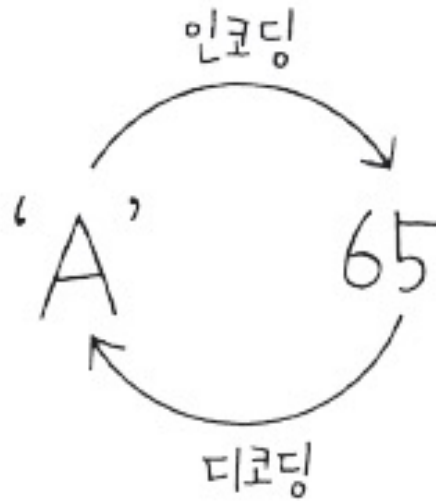
long

- 8바이트 자료형
- 가장 큰 정수 자료 형
- 숫자의 뒤에 L 또는 l 을 써서 long 형임을 표시해야 함
- 예) `int num = 12345678900; //오류` 남 int 의 범위 넘는 값 대입
 `long num = 12345678900; // 오류` 남
 숫자(리터럴) 12345678900 은 기본형이 int 인데 int 의 범위가 넘는 수
 => 숫자(리터럴) 12345678900을 long으로 처리하도록 명시
 `long num = 12345678900L; // ok` 소문자 l을 써도 되지만
 //1 과 구분하기 위해 대문자로 씀

•

char - 문자 자료형

- 컴퓨터에서는 문자도 내부적으로는 비트의 조합으로 표현
- 자바에서는 문자를 2 바이트로 처리
- 인코딩 - 각 문자에 따른 특정한 숫자 값(코드 값)을 부여
- 디코딩 - 숫자 값을 원래의 문자로 변환



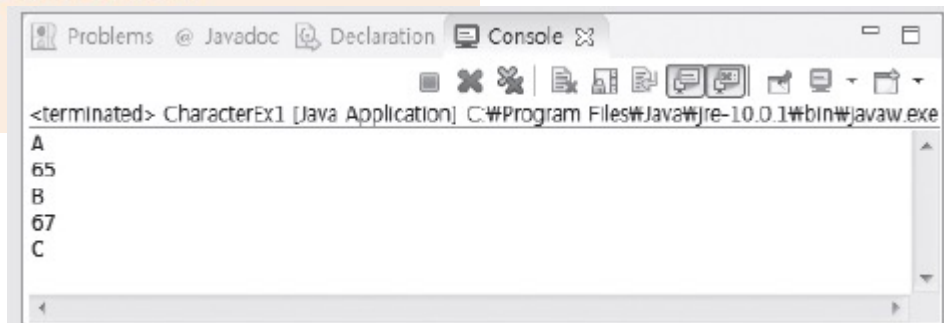
char 사용하기

```
package chapter2;

public class CharacterEx1 {
    public static void main(String[ ] args) {
        char ch1 = 'A';
        System.out.println(ch1);           // 문자 출력
        System.out.println((int)ch1);      // 문자에 해당하는 정수 값(아스키 코드 값) 출력

        char ch2 = 66;                     // 정수 값 대입
        System.out.println(ch2);           // 정수 값에 해당하는 문자 출력

        int ch3 = 67;
        System.out.println(ch3);           // 문자 정수 값 출력
        System.out.println((char)ch3);     // 정수 값에 해당하는 문자 출력
    }
}
```



The screenshot shows a Java IDE window with a console tab. The title bar indicates the application is "CharacterEx1 [Java Application]" and the path is "C:\Program Files\Java\jre-10.0.1\bin\javaw.exe". The console output displays the results of the program execution:

```
<terminated> CharacterEx1 [Java Application] C:\Program Files\Java\jre-10.0.1\bin\javaw.exe
A
65
B
67
C
```

문자 세트

- 문자세트 : 문자를 위한 코드 값 (숫자 값) 들을 정해 놓은 세트
- 아스키(ASCII) : 1 바이트로 영문자, 숫자, 특수문자 등을 표현함
- 유니코드 (Unicode) : 한글과 같은 복잡한 언어를 표현하기 위한
 - 표준 인코딩 UTF-8, UTF-16 이 대표적
 - (<https://www.unicode.org/charts/PDF/UAC00.pdf> 참고)
- 문자를 변수에 저장하면? 문자에 해당하는 코드 값이 저장됨
- 자바는 유니코드 UTF-16 인코딩 사용 함

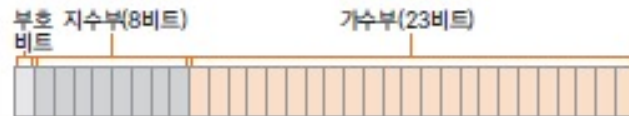
float, double - 실수 자료형

- 부동 소수점 방식: 실수를 지수부와 가수부로 표현함
무한의 실수를 표현하기 위한 방식
- 0.1 을 표현하는 방식

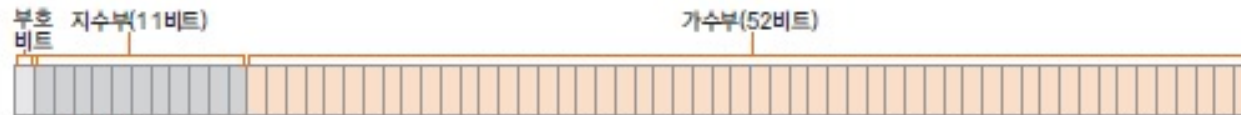
$$\begin{array}{ccc} \text{가수} & & \text{지수} \\ \boxed{1.0} \times \boxed{10}^{\boxed{-1}} \\ & & \text{밑수} \end{array}$$

밑수는 2, 10, 16 등을 주로 사용합니다.

- 실수 자료형 : float(4바이트) double(8바이트)



float형



double형

float, double - 실수 자료형

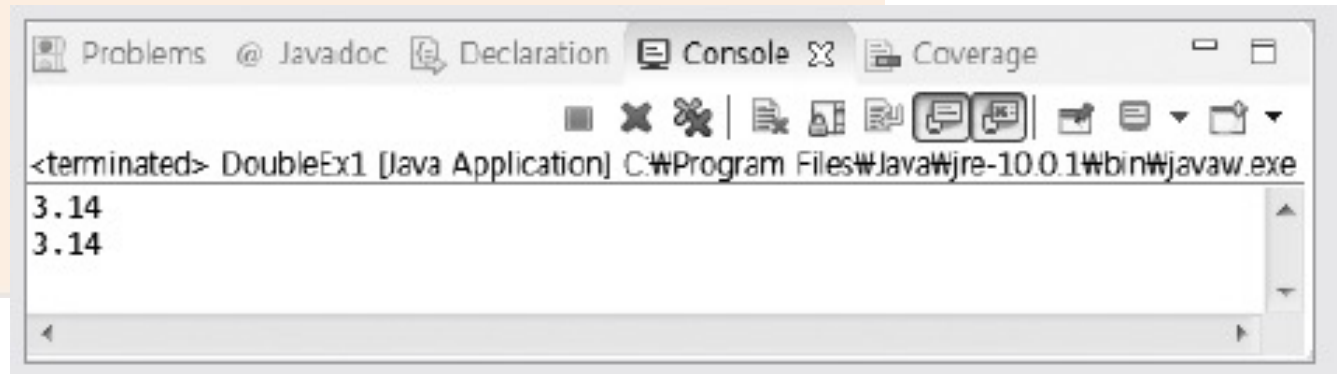
- 실수는 기본 적으로 double 로 처리 함
- float 형으로 사용하는 경우 숫자에 f, F 를 명시 함

```
package chapter2;

public class DoubleEx1 {
    public static void main(String[] args) {
        double dnum = 3.14;
        float fnum = 3.14F;

        System.out.println(dnum);
        System.out.println(fnum);
    }
}
```

식별자



부동 소수점 방식의 오류

- 지수와 가수로 표현 되는 부동 소수 점은 0을 표현할 수 없음
- 따라서 부동 소수점 방식에서는 약간의 오차가 발생할 수 있음

```
package chapter2;
```

```
public class DoubleEx2 {
```

```
    public static void main(String[ ] args) {
```

```
        double dnum = 1;
```

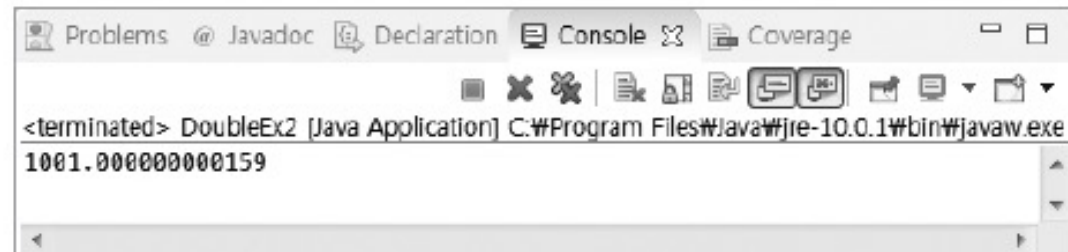
```
        for(int i = 0; i < 10000; i++) {  
            dnum = dnum + 0.1;  
        }
```

```
        System.out.println(dnum);
```

```
    }
```

```
}
```

for문은 지정한 문장을 정해진 횟수만큼 반복해서 수행하는 반복문입니다. 여기에서는 더하기를 10,000번 반복하라는 의미입니다. '04장 제어 흐름 이해하기'에서 자세히 배웁니다.



- 결과 값이 1001이 아님
- 오차를 감수하더라도 넓은 범위의 수를 표현하기 위해 사용

boolean - 논리형

- 논리값 true (참) , false(거짓) 을 표현하는 자료형
- boolean 으로 선언

```
package chapter2;

public class BooleanEx {
    public static void main(String[ ] args) {
        boolean isMarried = true;    //boolean 변수를 선언하고 초기화
        System.out.println(isMarried);
    }
}
```


자료형 없이 변수 사용하기 (자바 10)

- **자료형이 필요한 이유:**

- 변수를 선언 할 때는 변수가 사용할 메모리 크기와 타입을 구분하기 위해 자료형을 사용

- **지역 변수 자료형 추론 (local variable type inference) :**

- 변수에 대입되는 값을 보고 컴파일러가 추론

```
var num = 10;  
var dNum = 10.0;  
var str = "hello";
```



```
int num = 10;  
double dNum = 10.0;  
String str = "hello";
```

2-4. 상수

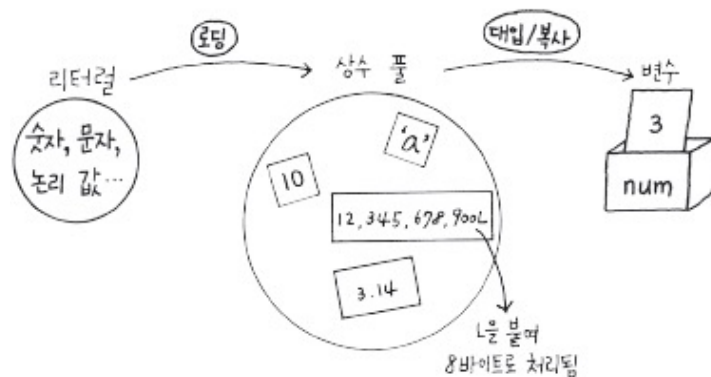
- 상수 : 변하지 않는 값 (cf 변수 : 변하는 값)
- 상수를 선언 : `final` 키워드 사용
- **final** double PI = 3.14;
- **final** int MAX_NUM = 100;

- `final` 로 선언된 상수는 다른 값을 대입 할 수 없음
- `PI = 3.15; // 에러 남`

- 프로그램 내에서 변경되지 말아야 하는 값을 상수로 선언 해 두고 혹시 변경되는 경우 선언된 값만 수정

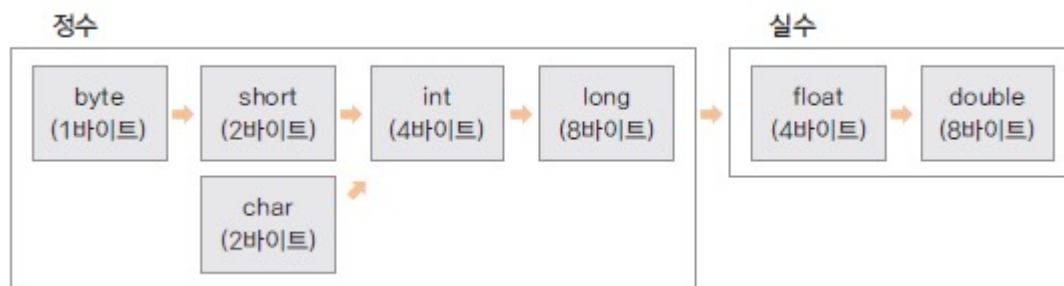
2-4. 리터럴(literal)

- 리터럴 : 프로그램에서 사용하는 모든 숫자, 값, 논리 값
- 예) 10, 3.14, 'A', true
- 리터럴에 해당되는 값은 특정 메모리 공간인 상수 풀(constant pool)에 있음
- 필요한 경우 상수 풀에서 가져와서 사용
- 상수 풀에 저장 할 때 정수는 int로 실수는 double로 저장
- 따라서 long 이나 float 값으로 저장해야 하는 경우 식별자 (L, I, F, f)를 명시해야 함



2-5. 형 변환(type conversion)

- 자료형은 각각 사용하는 메모리 크기와 방식이 다름
- 서로 다른 자료형의 값이 대입되는 경우 형 변환이 일어 남
- 묵시적 형변환 : 작은 수 에서 큰 수로
- 덜 정밀한 수에서 더 정밀한 수로 대입되는 경우



- 예) `long num = 3;` // int 값에서 long으로 자동 형 변환
- // L, l 을 명시하지 않아도 됨
- 명시적 형 변환: 묵시적 형 변환의 반대의 경우
- 변환 되는 자료 형을 명시해야 함 자료의 손실이 발생 할 수 있
- 음
- 예) `double dNum = 3.14;`
- `int num = (int)dNum;` //자료형 명시

감사합니다.

끝