# 2. 변수(Variable)와 자료형(Type)

# 변수(Variable)(1)

- 값을 저장할 수 있는 메모리의 특정 주소를 지칭하는 이름

```
int num;    // 변수 선언문 → int : 변수의 타입   num : 변수 이름
num = 100;   // 변수에 값 대입(저장)

int var = 10;   // 변수 선언과 함께 값을 대입(초기화)

int id, age;

int id = 1, age = 25;
id = age;    // 변수 간 값 대입
```

**[ Ex1 ] Variable.java**

```
package chap02.variable;

public class Variable {

	public static void main(String[] args) {
	// TODO Auto-generated method stub
		int a, b = 10;
		a = b;
		System.out.println("a의 값은 " + a + " 입니다.");
		System.out.println("b의 값은 " + b + " 입니다.");
	}
}
```

# Swap

- 변수 간 값 교환(Swap)

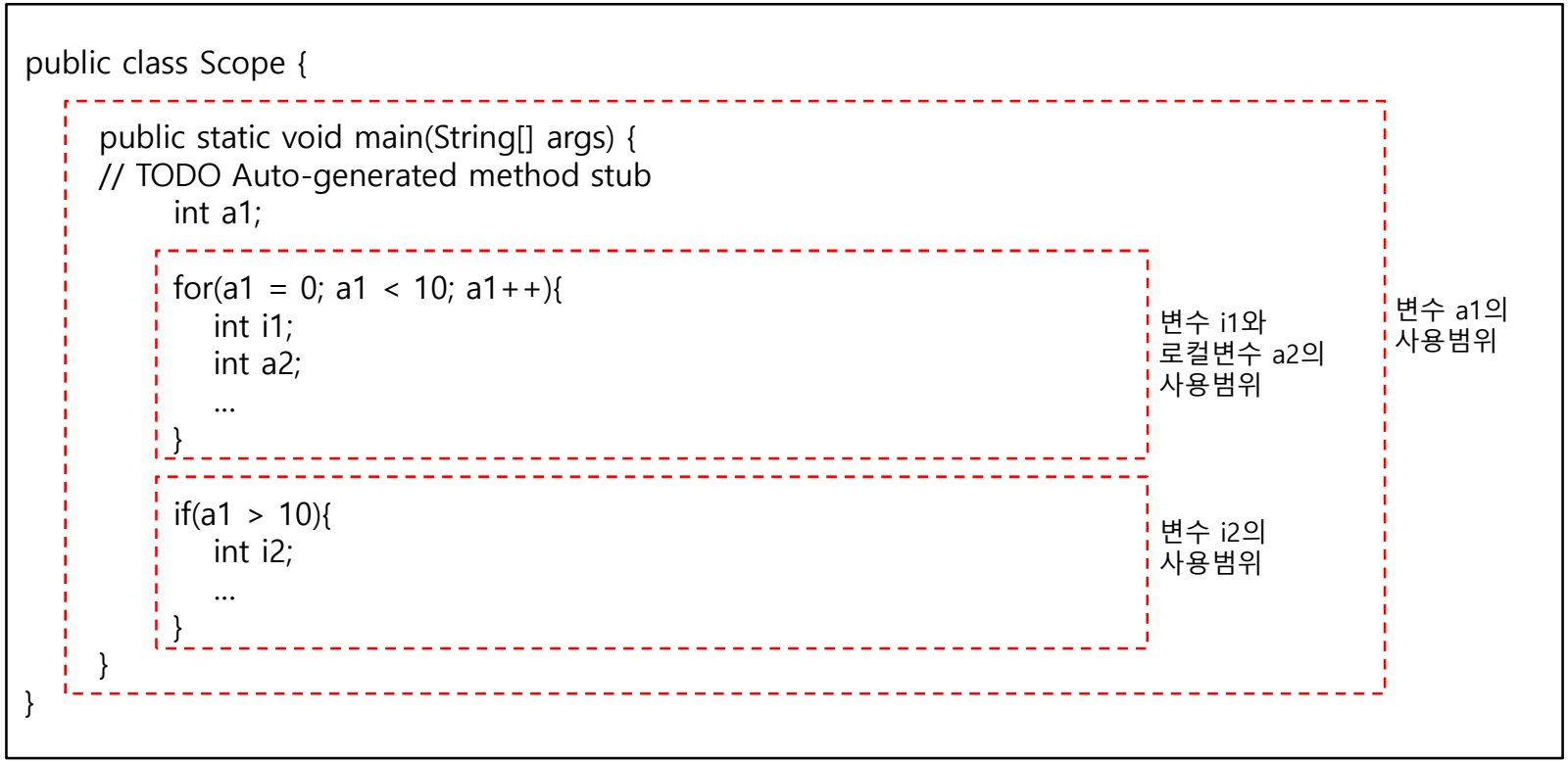
**[ Ex2 ] Swap.java**

```
package chap02.variable;

public class Swap {

	public static void main(String[] args) {
	// TODO Auto-generated method stub
		int a = 10, b = 20;
		int tmp;

		tmp = a;
		a = b;
		b = tmp;
		System.out.println(“a의 값은 ” + a + ” 입니다.”);
		System.out.println("b의 값은 " + b + " 입니다.");
	}
}
```

# Scope

- 변수 사용 범위(Scope)

  - 모든 변수는 중괄호{} 블록을 기준으로 사용 범위가 제한됨
    - 특정 메소드 블록 안에서 사용되는 변수를 로컬 변수라고 지칭



# 실습

**[ Ex3 ] Scope.java**

```
package chap02.variable;

public class Scope {
    public static void main(String[] args) {
    // TODO Auto-generated method stub
        int sum = 0;

        for(int i = 1; i <= 10; i++) {
            sum = sum + i;
        }
        System.out.println("1부터 10까지의 합은 " + sum + " 입니다.");
        // System.out.println("i의 값은 " + i + " 입니다."); // 에러 발생
        if(sum > 50) {
            int var = 50;
            System.out.println("sum의 값은 " + var + " 보다 큽니다.");
        }
        // System.out.println("sum의 값은 " + var + " 보다 큽니다."); // 에러 발생
        {
            // int sum = 30; // 중복 선언 에러
            int var2 = 100;
            System.out.println("lVal의 값은 " + var2 + " 입니다.");
        }
        // System.out.println("lVal의 값은 " + var2 + " 입니다."); // 에러 발생

        int var2 = 50;
        System.out.println("var2의 값은 " + var2 + " 입니다.");
    }
}
```

# 변수 네이밍(5)

- 변수 명명 규칙

  - 변수명은 변수가 사용되는 범위 안에서 unique 해야 함.
  - 대소문자가 구분되며 길이의 제한은 없음.
  - 숫자로 시작해서는 안 됨.
  - 특수문자는 '_'와 '$'만을 허용함.
  - 예약어를 사용해서는 안 됨.

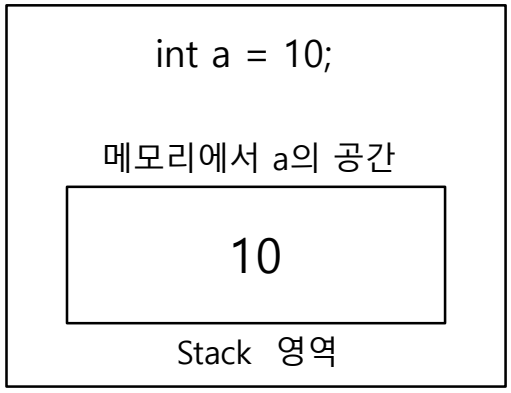
**[ 예약어 ]**

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

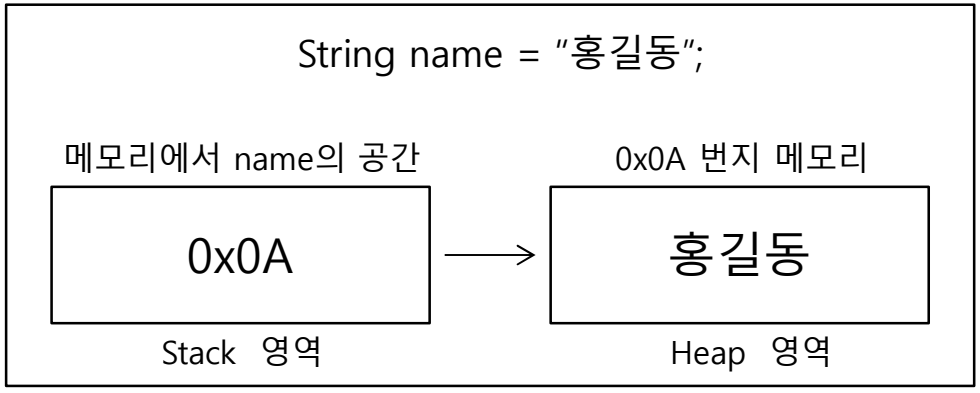
# 자료형

- 값을 저장하는 방식에 따라 기본형과 참조형으로 우선 분류

| 저장 방식에 따른 분류 | 저장되는 값에 따른 분류 | 타입 |
|---|---|---|
| 기본형 | 정수형 | byte, short, int, long |
| | 실수형 | float, double |
| | 문자형 | char |
| | 논리형 | boolean |
| 참조형 | 객체가 저장된 메모리 번지를 참조하는 타입<br>ex) 위의 8가지 기본형 타입을 제외한 나머지(배열, 스트링, 객체 등) | |



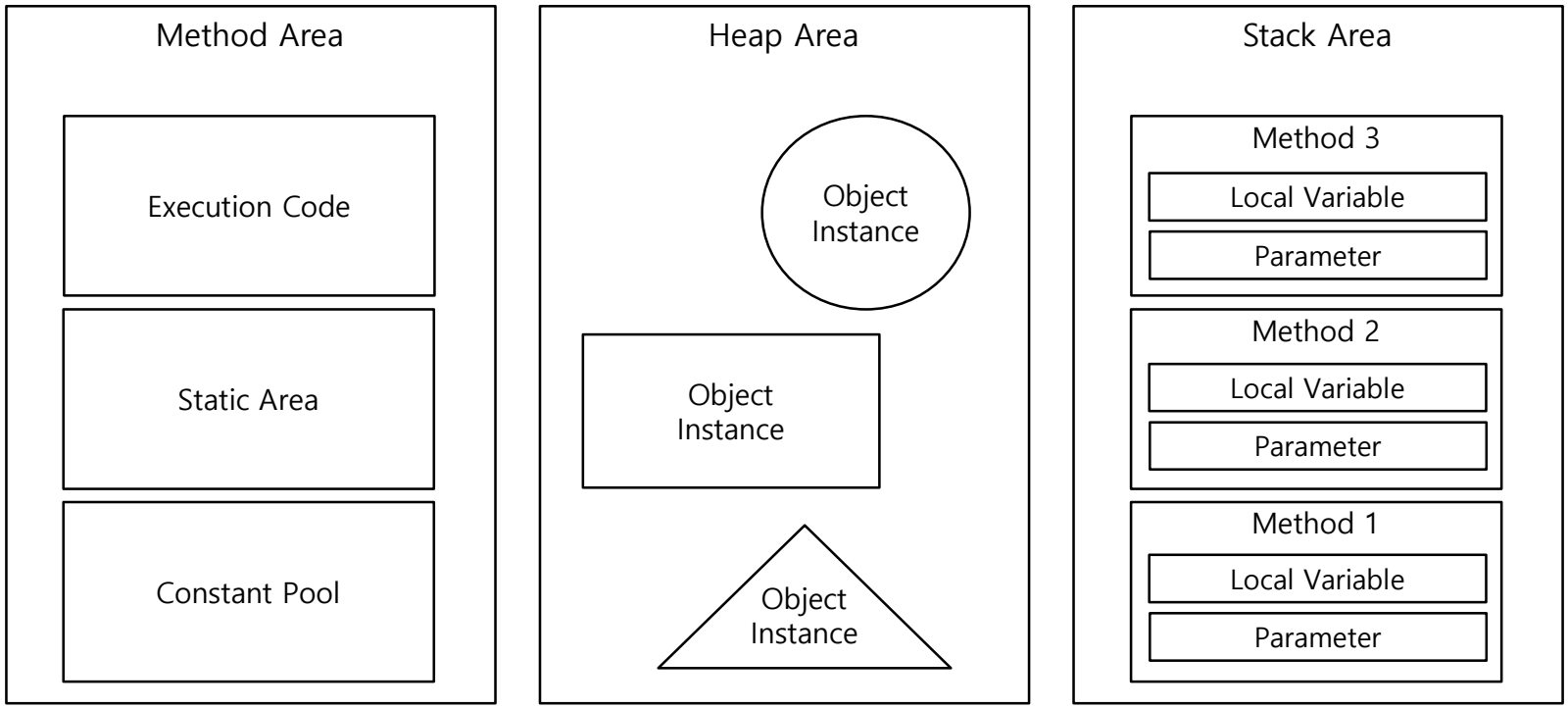

기본형




참조형

# 자바 메모리 구조

- 자바 런타임 메모리 구조



# 기본형

- 기본형(Primitive Type)

- 정수형

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

- 실수형

| 타입 | 크기 | | 값의 허용 범위(양수 기준) | 정밀도(소수점 이하 자리) |
|---|---|---|---|---|
| float | 4byte | 32bit | $(1.4x10^{-45})$ ~ $(3.4x10^{38})$ | 7자리 |
| double | 8byte | 64bit | $(4.9x10^{-324})$ ~ $(1.8x10^{308})$ | 15자리 |

- 문자형

| 타입 | 크기 | | 값의 허용 범위 | |
|---|---|---|---|---|
| char | 2byte | 16bit | 0 ~ $(2^{16}-1)$ | 0 ~ 65,535 (유니코드 값으로 저장) |

- 논리형

| 타입 | 크기 | | 값의 허용 범위 |
|---|---|---|---|
| boolean | 1byte | 8bit | true / false |

# 상수와 리터럴

- 상수(Constant)

  - 변수처럼 값을 저장하는 메모리 공간이지만 한번 저장한 값을 바꿀 수 없음
  - 선언 시, 자료형 앞에 final 키워드를 붙임

```
final int CURRENT_YEAR = 2024;
final float PI = 3.14f;
```

- 리터럴(Literal)

  - 변수에 저장하는 값의 자료형에 대한 표기법

[ 정수 리터럴 ]

```
int dec = 10;  // 10진수 10을 의미
int bin = 0b1010  // '0b'로 시작하며 2진수 1010을 의미 (10진수로 10)
int oct = 012  // 숫자 '0'으로 시작하며 8진수 12을 의미 (10진수로 10)
int hex = 0xA  // '0x'로 시작하며 16진수 A를 의미(10진수로 10)
```

[ 실수 리터럴 ]

```
double dblVar = 3.14;  // double 형식의 값을 표현
float fltVar = 3.14f;  // float 형식의 값을 표현
```

[ 논리 리터럴 ]

```
boolean blnVar = true;  // 참을 의미
```

[ 문자 및 문자열 리터럴 ]

```
char chrVar = 'A';  // 하나의 문자를 나타낼 경우, ' '를 사용
String strVal = "ABC";  // 문자열을 나타낼 경우, " "를 사용
```

# 실습(1)

**[ Ex4 ] IntLiteral.java**

```java
package chap02.type;

public class IntLiteral {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int binVal = 0b1011;
        int octVal = 0101;
        int decVal = 101;
        int hexVal = 0x101;

        System.out.println("binVal : " + binVal);
        System.out.println("octVal : " + octVal);
        System.out.println("decVal : " + decVal);
        System.out.println("hexVal : " + hexVal);
    }
}
```

# 실습(2)

**[ Ex5 ] Constant.java**

```java
package chap02.type;

public class Constant {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        final float PI = 3.14f;
        double pi = 3.141592;
        //float pi2 = 3.141592; // 에러

        //PI = pi; // 에러

        System.out.println("원주율 PI는 " + PI + " 입니다.");
        System.out.println("원주율 PI는 " + pi + " 입니다.");

        int r = 10;
        System.out.println("반지름이 10인 원의 넓이는 " + r*r*PI + " 입니다.");
    }
}
```

# 실습(3)

**[ Ex6 ] Char.java**

```java
package chap02.type;

public class Char {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        char chrA = 'A';
        System.out.println("chrA는 " + chrA + " 입니다.");

        char ascA = 65;
        System.out.println("ascA는 " + ascA + " 입니다.");

        char uniA = '\u0041'; // 유니코드로 16진수 0041 문자를 의미
        System.out.println("uniA는 " + uniA + " 입니다.");

        char chrGa = '가';
        System.out.println("chrGa는 " + chrGa + " 입니다.");

        char uniDecGa = 44032;
        System.out.println("uniDecGa는 " + uniDecGa + " 입니다.");

        char uniHexGa = '\uac00'; // 유니코드로 16진수 ac00 문자를 의미
        System.out.println("uniHexGa는 " + uniHexGa + " 입니다.");
    }
}
```

# 형변환(1)

- 형변환(Type Casting)

  - 변수 또는 상수의 타입을 다른 타입으로 변환하는 것

  - 자동 형변환

    • 값의 허용 범위가 작은 타입의 값을 허용 범위가 큰 타입으로 저장할 때 발생

```
byte a = 10;
int b = a;   // 자동 형 변환 → int b = (int)a;
```

```
float fVal = 0.1234f;
double dVal = fVal;   // 자동 형 변환 → double dVal = (double)fVal;
```

```
char chrVal = 'A';  //
int iVal = chrVal;   // 65가 저장됨

char chrValGa = '가';
int iValGa = chrValGa;  // 44032가 저장됨
```

# 형변환(2)

- 강제 형변환
  - 값의 허용 범위가 큰 타입의 값을 허용 범위가 작은 타입으로 저장할 때 발생

```
int iVal = 10;
byte bVal = (byte)iVal;   // int 타입의 iVal을 byte 타입으로 강제 형변환
```

```
double dVal = 0.1234;
float fVal = (float)dVal;   // double 타입의 dVal을 float 타입으로 강제 형변환
```

```
int iVal = 65;
char chrVal = (char)iVal;   // 'A'가 저장됨

int iValGa = 44032;
char chrValGa = (char)iValGa;  // '가'가 저장됨
```

# 실습(1)

**[ Ex7 ] TypeCasing.java**

```java
package chap02.type;

public class TypeCasting {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        byte a = 10;
        int b = a;
        System.out.println("byte a를 int b로 형변환 : " + b);

        float fVal = 0.1234f;
        double dVal = fVal;
        System.out.println("float fVal을 double dVal로 형변환" + dVal);

        char chrValGa = '가';
        int iValGa = chrValGa;

        System.out.println("iValGa의 값은 " + iValGa + " 입니다.");

        // 허용범위가 같은 타입끼리는 자동형변환이 되지 않음
        char chrVal = 'A';
        byte bVal = (byte)chrVal;
        System.out.println("bVal의 값은 " + bVal + " 입니다.");
        byte byteValue = 65;
        char charValue = (char)byteValue;
        System.out.println(charValue);
    }
}
```

# 실습(2)

**[ Ex8 ] TypeCasing2.java**

```java
package chap02.type;

public class TypeCasting2 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int iVal = 65;
        byte bVal = (byte)iVal;    // int 타입의 iVal을 byte 타입으로 강제 형변환
        System.out.println("bVal은 " + bVal + " 입니다.");

        double dVal = 0.1234;
        float fVal = (float)dVal;    // double 타입의 dVal을 float 타입으로 강제 형변환
        System.out.println("fVal은 " + fVal + " 입니다.");

        char chrVal = (char)iVal;    // 'A'가 저장됨
        System.out.println("chrVal은 " + chrVal + " 입니다.");

        int iValGa = 44032;
        char chrValGa = (char)iValGa;  // '가'가 저장됨
        System.out.println("chrValGa는 " + chrValGa + " 입니다.");

        iVal = 300;
        bVal = (byte)iVal; // 마지막 한 바이트만 저장되어 값 손실 발생
        System.out.println("bVal은 " + bVal + " 입니다.");
    }
}
```

# 정수 연산 시 자동 형변환(1)

- 정수 연산에서의 자동 형변환

  - int보다 허용범위가 작은 변수(byte, char, short)가 산술 연산식에서 피연산자로 사용되면 int 타입으로 자동 형변환

```
byte x = 10;
byte y = 20;

int result = x + y;  // x와 y가 각각 int형으로 형변환 된 이후에 연산이 이루어짐
```

  - int보다 허용범위가 큰 변수(long 타입)가 산술 연산식에서 피연산자로 사용되면 다른 피연산자는 무조건 long 타입으로 자동 형변환되어 연산이 수행됨

```
byte x = 10;
int y = 20;
long z = 30;

long result = x + y + z;  // x와 y가 각각 long형으로 형변환 된 이후에 연산이 이루어짐
```

# 실수 연산 시 자동 형변환(2)

- 실수 연산에서의 자동 형변환

  - double 타입의 피연산자와 함께 연산을 하는 다른 피연산자는 무조건 double로 자동 형변환

```
float x = 10.01f;
double y = 20.35;

double result = x + y;  // x가 double로 자동 형변환
```

  - 정수형 피연산자도 double 타입의 피연산자와 연산을 하면 double로 자동 형변환

```
int iVal = 100;
double dVal = 20.35;

double result = iVal + dVal;  // iVal이 double로 자동 형변환
```

  - 정수로 계산하려면 강제 형변환을 통해 double을 int로 변환(이 때, 소수점 이하는 버려짐)

```
int iVal = 100;
double dVal = 20.35;

int result = iVal + (int)dVal;  // dVal이 int로 강제 형변환되어 dVal = 20으로 계산됨
```

  - 나눗셈 연산을 할 경우, 자동으로 double로 형변환

```
int iVal1 = 10;
int iVal2 = 5;

double result = iVal1 / iVal2;  // double로 강제 형변환되어 계산결과는 2.0
```

# 실습

**[ Ex9 ] TypeCasingOp.java**

```java
package chap02.type;

public class TypeCastingOp {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        byte x = 10;
        byte y = 20;
        int result = x + y; // x와 y가 모두 int로 형변환
        System.out.println("result는 " + result + " 입니다.");
        //byte xyResult = x + y; // 에러 --> X와 y가 int로 형변환 되었으므로 byte 타입에 결과를 넣을 수 없음
        byte xyResult = (byte)(x + y); // 계산 결과를 byte로 강제 형변환

        float x1 = 10.01f;
        double y1 = 20.35;
        double result1 = x1 + y1;  // x가 double로 자동 형변환
        System.out.println("result1는 " + result1 + " 입니다.");

        int iVal = 100;
        double dVal = 20.35;
        double result2 = iVal + dVal;  // iVal이 double로 자동 형변환
        System.out.println("result2는 " + result2 + " 입니다.");

        int result3 = iVal + (int)dVal;  // dVal이 int로 강제 형변환되어 20으로 계산됨
        System.out.println("result3는 " + result3 + " 입니다.");

        int iVal1 = 10;
        int iVal2 = 5;

        double result4 = iVal1 / iVal2;  // double로 강제 형변환되어 계산결과는 2.0
        System.out.println("result4는 " + result4 + " 입니다.");

        iVal2 = 4;
        int result5 = iVal1 / iVal2;  // int로 강제 형변환되어 계산결과는 2
        System.out.println("result4는 " + result5 + " 입니다.");

        //int result6 = iVal1 / 4.0; // 에러
        int result6 = (int)(iVal1 / 4.0);
    }
}
```

# 문자열 연산 자동 형변환

- String 자동 형변환

  - 피연산자 중 하나가 문자열(String)인 경우, 다른 피연산자도 자동으로 문자열로 형변환

```
String str = "5" + 8;   // "58"

String str = 5 + "8";   // "58"
```

  - 일반적인 사칙연산과 마찬가지로 왼쪽부터 계산

```
String str = 3 + 5 + "8";  // "88"

String str = 3 + "5" + 8;  // "358"

String str = "3" + 5 + 8;  // "358"
```

# 실습

**[ Ex10 ] StringConcat.java**

```java
package chap02.type;

public class StringConcat {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String str1 = "5" + 8;
        System.out.println("str1은 \"" + str1 + "\" 입니다.");

        String str2 = 5 + "8";
        System.out.println("str2은 \"" + str2 + "\" 입니다.");

        String str3 = 3 + 5 + "8";
        System.out.println("str3은 \"" + str3 + "\" 입니다.");

        String str4 = 3 + "5" + 8;
        System.out.println("str4은 \"" + str4 + "\" 입니다.");

        String str5 = "3" + 5 + 8;
        System.out.println("str5은 \"" + str5 + "\" 입니다.");

        String str6 = "3" + (5 + 8);
        System.out.println("str6은 \"" + str6 + "\" 입니다.");

        String fName = "길동";
        String lName = "홍";
        String Name = lName + fName;
        System.out.println("제 이름은 \"" + Name + "\" 입니다.");
    }
}
```

# 문자열 강제 형변환

- 문자열을 기본 타입으로 강제 형변환

- String을 byte 타입으로 변환

```
byte bVal = Byte.parseByte("15");
```

- String을 short 타입으로 변환

```
short sVal = Short.parseShort("15");
```

- String을 int 타입으로 변환

```
int iVal = Integer.parseInt("15");
```

- String을 long 타입으로 변환

```
long lVal = Long.parseLong("15");
```

- String을 float 타입으로 변환

```
float fVal = Float.parseFloat("15.235");
```

- String을 double 타입으로 변환

```
double dVal = Double.parseDouble("15.235");
```

- String을 boolean 타입으로 변환

```
boolean bVal = Boolean.parseBoolean("true");
```

# 실습

**[ Ex11 ] StringToPrimitive.java**

```java
package chap02.type;

public class StringToPrimitive {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String intStr = "15";
        String doubleStr = "15235";
        String boolStr = "true";

        System.out.println("intStr의 타입은 " + ((Object)intStr).getClass().getName() + " 입니다.");
        System.out.println("doubleStr의 타입은 " + ((Object)doubleStr).getClass().getName() + " 입니다.");
        System.out.println("boolStr의 타입은 " + ((Object)boolStr).getClass().getName() + " 입니다.");

        byte bVal = Byte.parseByte(intStr);
        System.out.println("bVal의 타입은 " + ((Object)bVal).getClass().getName() + " 입니다.");
        short sVal = Short.parseShort(intStr);
        System.out.println("sVal의 타입은 " + ((Object)sVal).getClass().getName() + " 입니다.");
        int iVal = Integer.parseInt(intStr);
        System.out.println("iVal의 타입은 " + ((Object)iVal).getClass().getName() + " 입니다.");
        long lVal = Long.parseLong(intStr);
        System.out.println("lVal의 타입은 " + ((Object)lVal).getClass().getName() + " 입니다.");
        float fVal = Float.parseFloat(doubleStr);
        System.out.println("fVal의 타입은 " + ((Object)fVal).getClass().getName() + " 입니다.");
        double dVal = Double.parseDouble(doubleStr);
        System.out.println("dVal의 타입은 " + ((Object)dVal).getClass().getName() + " 입니다.");
        boolean boolVal = Boolean.parseBoolean(boolStr);
        System.out.println("boolVal의 타입은 " + ((Object)boolVal).getClass().getName() + " 입니다.");
    }
}
```

# 표준 출력

- 변수를 표준출력 시스템으로 내보내기

- 표준 출력

  - System.out.println

```
System.out.println("문자열");
System.out.println("문자열" + var);
System.out.println(var);

System.out.print("문자열");  // 사용법은 println과 같으나 줄바꿈을 하지 않음
```

  - System.out.printf

```
String name = "홍길동";
int age = 35;

System.out.printf("나는 %s이고 %d살 입니다.", name, age);
```

| 지시자 | 예 |
|---|---|
| %d | 10진 정수 형식으로 출력 |
| %f | 부동 소수점 형식으로 출력 |
| %s | 문자열로 출력 |
| %c | 문자로 출력 |
| %o | 8진 정수의 형식으로 출력 |

| 지시자 | 예 |
|---|---|
| %x, %X | 16진 정수의 형식으로 출력 |
| %b | 불리언 형식으로 출력 |
| %e, %E | 지수 표현식 형태로 출력 |
| ₩t | 탭(tab) 출력 |
| ₩n | 줄 바꿈 |
| %% | % 출력 |

# 실습(1)

**[ Ex12 ] PrintEx.java**

```java
package chap02.type;

public class PrintfEx {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        byte b = 1;
        short s = 2;
        char c = 'A';

        int finger = 10;
        long big = 100_000_000_000L;
        long hex = 0xFFFF_FFFF_FFFF_FFFFL;  // long hex = 0xFFFFFFFFFFFFFFFFL;

        int octNum = 010;  // 8진수 10, 10진수로는 8
        int hexNum = 0x10; // 16진수 10, 10진수로는 16
        int binNum = 0b10; // 2진수 10, 10진수로는 2

        System.out.printf("b = %d\n", b);
        System.out.printf("s = %d\n", s);
        System.out.printf("c = %c, %d \n", c, (int)c);
        System.out.printf("finger = [%5d]\n", finger);
        System.out.printf("finger = [%-5d]\n", finger);
        System.out.printf("finger = [%05d]\n", finger);
        System.out.printf("big = %d\n", big);
        System.out.printf("hex = %#x\n", hex);  // '#'은 접두사(16진수 0x, 8진 0)
        System.out.printf("octNum = %o, %d\n", octNum, octNum);
        System.out.printf("hexNum = %x, %d\n", hexNum, hexNum);
        System.out.printf("binNum = %s, %d\n", Integer.toBinaryString(binNum), binNum);
    }
}
```

# 실습(2)

**[ Ex13 ] PrintEx2.java**

```java
package chap02.type;

public class PrintEx2 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String url = "www.naver.com";

        float f1 = .10f;    // 0.10, 1.0e-1
        float f2 = 1e1f;    // 10.0, 1.0e1, 1.0e+1
        float f3 = 3.14e3f;
        double d = 1.23456789;

        System.out.printf("f1=%f, %e, %g%n", f1, f1, f1);
        System.out.printf("f2=%f, %e, %g%n", f2, f2, f2);
        System.out.printf("f3=%f, %e, %g%n", f3, f3, f3);

        System.out.printf("d=%f%n", d);
        System.out.printf("d=%14.10f%n", d); // 전체 14자리 중 소수점 10자리
        System.out.printf("d=%014.10f%n", d); // 전체 14자리 중 소수점 10자리, 왼쪽 공백은 0으로 채움

        System.out.printf("[%s]%n", url);
        System.out.printf("[%20s]%n", url);
        System.out.printf("[%-20s]%n", url); // 오른쪽 정렬
        System.out.printf("[%.8s]%n", url);  // 왼쪽에서 8글자만 출력
    }
}
```

# 표준 입력

- 표준입력 시스템으로부터 입력 받기

  - 표준 입력

    - System.in.read()

      입력 문자 뒤에 CR(Carriage Return - Ascii 13)과 LF(Line Feed – Ascii 10)이 따라옴

```
int keyCode1 = System.in.read();   // keyCode1에 키보드에서 입력한 문자가 저장
int keyCode2 = System.in.read()    // keyCode2에 13 저장
int keyCode3 = System.in.read()    // keyCode3에 10 저장
```

    - Scanner

      행 단위로 입력을 받을 수 있음

```
Import java.util.Scanner;

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

# 실습(1)

**[ Ex14 ] KeyCode.java**

```
package chap02.type;

public class KeyCode {
    public static void main(String args[]) throws Exception {
        int key;

        while(true) {
            key = System.in.read();
            System.out.println("input key: " + key);

            if(key == 'q')
                break;
        }
        System.out.println("quit!!");
    }
}
```

# 실습(2)

**[ Ex15 ] InputScanner.java**

```java
package chap02.type;

import java.util.Scanner;

public class InputScanner {
    public static void main(String[] args) throws Exception {
        Scanner scanner = new Scanner(System.in);
        String in;

        while(true) {
            in = scanner.nextLine();
            System.out.println("in : " + in);
            if(in.equals("quit")) { //
                break;
            }
        }
        System.out.println("quit!!");
    }
}
```