Lesson 5: Number and Operators

The assignment operator:

The assignment operator is the standard equal sign (**=**) and is used to "assign" a value to a variable.

```
int i = 3;        // Ok,… assign the value 3 to i. Notice the direction of data flow.

3 = i;            // Illegal! Data never flows this way!

double p;
double j = 47.2;
p = j;            // assign the value of j to p. Both p and j are now equal to 47.2
```

Multiple declarations:

It is possible to declare several variables on one line:

```
double d, mud, puma;                          //the variables are only declared
double x = 31.2, m = 37.09, zu, p = 43.917;   //x, m, & p declared and initialized
                                              // zu is just declared
```

Fundamental arithmetic operations:

The basic arithmetic operation are +, -, * (multiplication), / (division), and % (modulus).

Modulus is the strange one. For example, System.out.println(5%3); will print 2. This is because when 5 is divided by 3, the **remainder** is 2. **Modulus gives the remainder.** Modulus also handles negatives. The answer to *a*%*b* always has the same sign as *a*. The sign of *b* is ignored.

PEMDAS:

The algebra rule, PEMDAS, applies to computer computations as well. (PEMDAS stands for the order in which numeric operations are done. P = parenthesis, E = exponents, M = multiply, D = divide, A = add, S = subtract. Actually, M and D have equal precedence, as do A and S. For equal precedence operation, proceed from left to right. A mnemonic for PEMDAS is, "Please excuse my dear Aunt Sally"… See Appendix H for the precedence of all operators.)

```
System.out.println(5 + 3 * 4 –7);              //10
System.out.println(8 – 5*6 / 3 + (5 –6) * 3);  //-5
```

Not the same as in Algebra:

An unusual assignment….consider the following:

```
count = count +3;    //this is illegal in algebra; however, in computer science it
                     //means the new count equals the old count + 3.
int count =15;
count = count + 3;
```

```
System.out.println(count);      //18
```

Increment and Decrement:

The increment operator is ++, and it means to add one. The decrement operator is --, and it means to subtract one:

```
x++; means the same as      x = x +1;
x--; means the same as      x = x – 1;
x++ is the same as          ++x (the ++ can be on either side of x)
x-- is the same as          --x (the -- can be on either side of x)

int y = 3;
y++;
System.out.println(y);      //4
```

Compound operators:

Syntax Example Simplified meaning

a. +=
   x += 3; → x = x + 3;

b. -=
   x -= y - 2; → x = x – (y - 2);

c. *=
   z*= 46; → z = z * 46;

d. /=
   p/= x-z; → p = p / (x-z);

e. %=
   j%= 2 → j = j%2;

Code Examples

```
int g = 409;
g += 5;
System.out.println(g);      //414

double d = 20.3;
double m =10.0;
m*=d –1;
System.out.println(m);      //193.0
```

The whole truth:

Actually, the full truth was not told above concerning *x*++. It does not always have the same effect as does ++*x*. Likewise, *x*-- does not always have the same effect as does --*x*.

x++ increments *x* **after** it is used in the statement.

++*x* increments *x* **before** it is used in the statement.

Similarly,
x-- decrements *x* **after** it is used in the statement.
--*x* decrements *x* **before** it is used in the statement.

Code Examples

```
int q = 78;
int p = 2 + q++;
System.out.println("p = " + p + ", q = " + q);          //p = 80, q = 79

int q = 78;
int p = ++q + 2;
System.out.println("p = " + p + ", q = " + q);          //p = 81, q = 79
```

Integer division truncation:

When dividing two integers, the fractional part is truncated (thrown away) as illustrated by the following:

```
int x = 5;
int y = 2;
System.out.println(x / y); //Both x and y are integers so the "real" answer of 2.5
                           //has the fractional part thrown away to give 2
```