Java Programming

Arithmetic

Basic Math Operators

- Simple arithmetic in Java is fairly simple. There are five basic math operators. The first four of which you probably learned in elementary school. The fifth you may have learned in a math course in middle school. The basic math operators are:
  - `+` Addition
  - `-` Subtraction
  - `*` Multiplication
  - `/` Division
  - `%` Mod Division *(Modulus Operator)*
- Note: The division operator (/) is used for both decimal and integer division. The mod division operator gives the remainder of an integer division.

Simple Examples

```
double x = 4.0;     // Create variable x and set to 2.0
double y = 2.0;     // Create variable y and set to 3.0
double z;           // Create variable z
z = x + y;
System.out.println("z = " + z);     // This will print "z = 6.0"
z = x - y;
System.out.println("z = " + z);     // This will print "z = 2.0"
z = x * y;
System.out.println("z = " + z);     // This will print "z = 8.0"
z = x / y;
System.out.println("z = " + z);     // This will print "z = 2.0"

int a = 10;     // Create integer variable a and set to 10
int b = 7;      // Create integer variable b and set to 7
int c;          // Create integer variable c
c = a / b;      // This will do integer division
System.out.println("c = " + c);     // This will print "c = 1"
c = a % b;    // Do mod division to get remainder after a / b
System.out.println("c = " + c);     // This will print "c = 3"
```

Complex Example

```
z = x + y * 3.0 / 4.0 - 1.5
System.out.println("z = " + z);
```

- Would you expect to see the output as z = 3.0?
- The actual output would be z = 4.0.
- The problem here is a thing called precedence.
- Java does not add 4.0 and 2.0 then multiply by 3.0, divide that by 4.0, and finally subtract 1.5 to get the result.
- Java does all multiplication and division first moving from left to right.
- Then it does the addition and subtraction.
- Thus (4.0 + 2.0 * 3.0 / 4.0 - 1.5) becomes (4.0 + 1.5 - 1.5) after doing the multiplication and division and then this gives the final answer of 4.0.

- If you want the order to be different then you must use parentheses. Java always completes all calculations inside parentheses first. Thus the code…

```
z = ((x + y) * 3.0 / 4.0) - 1.5
System.out.println("z = " + z);
```

- …will give the desired output of **z = 3.0**.

Unary Operators

- Unary operators return the result of an operation on only one operand. There are six defined in the Java language:
  - **+** `Indicates positive value`
  - **-** `Negates an expression`
  - **++** `Increments a value by 1`
  - **--** `Decrements a value by 1`
  - **!** `Inverts the value of a boolean`
  - **~** `Inverts the bits of a binary number`
- The increment and decrement operators are special because they can be applied in both prefix (before the operand) and postfix (after the operand) positions.

Prefix Increment (++) Operator

- The following adds 1 to x and stores the value of x into y:

```
int x = 0;
int y = 0;
y = ++x;
```

- `x` has the value of `1`
- `y` has the value of `1`

Postfix Increment (++) Operator

- The following adds 1 to x and stores the value of x into y:

```
int x = 0;
int y = 0;
y = x++;
```

- `x` has the value of `1`
- `y` has the value of `0`

Assignment Operators

- **=** Assigns right variable value to left variable
- **+=** Adds left and right values and stores result in the left variable
- **-=** Subtracts left and right values and stores result in the left variable
- ***=** Multiplies left and right values and stores result in the left variable
- **/=** Divides left and right values and stores result in the left variable
- **%=** Mods left and right values and stores result in the left variable

Assignment Operator Examples

```
int z = 0; // z = 0

z -= 5;    // z = -5
z += 10;   // z = 5
z *= 4;    // z = 20
z /= 2;    // z = 10
z %= 3;    // z = 1
```

Assignment Operators with Increment & Decrement Operators

```
int x = 4;
int y = 2;
int z = 0;

z += x++;    // x = 5, y = 2, z = 4
z -= x--;    // x = 4, y = 2, z = -1
z /= --y;    // x = 4, y = 1, z = -1
z *= x++;    // x = 5, y = 1, z = -4
z %= ++x;    // x = 6, y = 2, z = 0
```

Higher Math Functions

- Java also has a number of built in higher math functions.
  - Trigonometry
  - Exponential
  - Logarithmic
  - etc.
- All of these functions are accessed by preceding the name of the function with **Math.**

Examples

```
// Create variable angDeg and set to 45 degrees
double angDeg = 45.0;

// Create variable angRad for angles in radians
double angRad;

// Create variable result
double result;


// Convert 45.0 degrees to radians
angRad = Math.toDegrees(angDeg);
System.out.println("45.0 degrees in radians = " + angRad );

// This will print the sine of the angle
result = Math.sin(angRad);
System.out.println("sin(" + angDeg + ") = " + result );

// This will print the cosine of the angle
result = Math.cos(angRad);
System.out.println("cos(" + angDeg + ") = " + result );

// This will print the tangent of the angle
result = Math.tan(angRad);
System.out.println("tan(" + angDeg + ") = " + result );

// This will print the square root of 45.0
result = Math.sqrt(angDeg);
System.out.println("sqrt(" + angDeg+ ") = " + result );
```

Math Constant Fields

- Math.E
  - The double value that is closer than any other to *e*, the base of the natural logarithms.
- Math.PI
  - The double value that is closer than any other to *pi*, the ratio of the circumference of a circle to its diameter.