Casting

General format:

`(cast type) sometype var;`

Example

`(int) double x=5.2;` - now x is 5

This is called truncating, and w hen a double get truncated, everything after the decimal gets chopped off ed. Thus, the variable looses its precision.

Another Example

a and b are ints, so `a/b` get truncated. In order to keep the decimal:

`int x = (double)a/b;`

Strings

Strings in double quotes ("") are recognized as literal constants or literal Strings.
That means w hatever you w rite betw een those tw o double quotes is exactly w hat your compiler is going not print out

'+' or '+=' (String concatenation). When you do this, the strings get put together

`"a"+"b"`

`"Catch"+22;` - "Catch 22"

Escape Sequences

`\n` - new line
`\r` - carriage return
`\t` - tab or indent
`\\` - backslash
`\"` - double quotes
`\'` - single quote

`System.out.print("I like tacos.\n I do too.");`
This prints:

```
I like tacos.
I do too.
```

Symbolic Constants

Initialized *final* variables

example: `private final int sidelength=8;`

easy to change the value throughout the program, if necessary

`private final double TAX_RATE;` if you every see a variable in all caps w ith underscores it is likely a constant

Arithmetic

Binary Operators

Operators: `+,-,/,*,%`
Order that they are carried out is the same as PEMDAS. In this case, modulus is has the same rank in PEMDAS as multiplication or division

Unary Operators

Operators: `-,!,++,--`
Have higher precedence than binary operators w hich means they are executed first. Example: `side=-side;`

Compound Operators

`a=a+b;` <-> `a+=b;`

`a=a-b;` <-> `a-=b;`

`a=a*b;` <-> `a*=b;`

`a=a/b;` <-> `a/=b;`

`a=a%b;` <-> `a%=b;`

The Math Class

`Math.abs(x)` - |x|

`Math.round(x)` - round to nearest int

`Math.pow(x, e)` - x^e

`Math.sqrt(x)` - square root of x

`Math.max(x, y)` - greater of the tw o

`Math.min(x, y)` - smaller of the tw o

`Math.random()` - 0<=x<1