

Level 2

Week 1

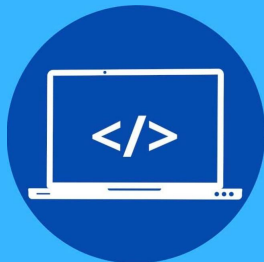


Hello World



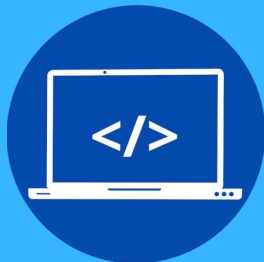
IDE / Web Interface

- Windows / Mac - Eclipse/JGrasp with the Java jdk file.
- Chromebook - Coding Playground



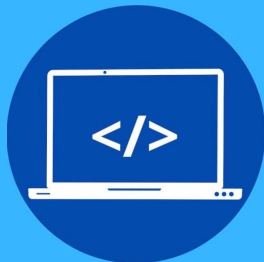
Programming Skeleton

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



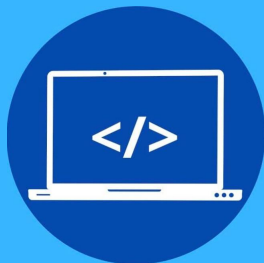
Adding Some Meaningful Code

```
public class Tester{ //We could put any name here besides tester  
  
    public static void main(String args[])  
  
    {  
  
        System.out.println("Hello world");  
  
    }  
  
}
```



Printing

1. "Peter Piper picked a peck of pickled peppers."
2. "I like computer science."
3. $25/5$
4. $4 / 7.0445902$
5. $13 * 159.56$



Two printlns for the price of one

```
public static void main(String args[])  
{  
System.out.println("Hello world");  
System.out.println("Hello again");  
}
```

Run this and note that it prints :

Hello world

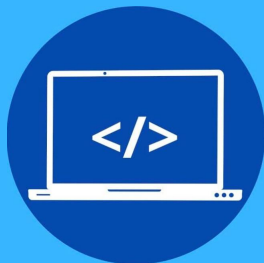
Hello again



Printing “Sideways”

```
public static void main(String args[])  
  
{  
  
System.out.print("Hello world");  
  
System.out.println("Hello again");  
  
}
```

Hello worldHello again



An in-depth look at remarks

```
public class Tester

{

//Programmer: Kosmo Kramer

//Date created: Sept 34, 1492

//School: Charles Manson High School; Berkley, Ca

public static void main(String args[])

{

System.out.println("Hello again");

}

}
```

```
public class Tester

{

/*Programmer: Kosmo Kramer

Date created: Sept 34, 1492

School: Charles Manson Junior High; Berkley, Ca*/

public static void main(String args[])

{

System.out.println("Hello again");

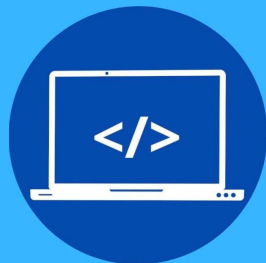
}

}
```



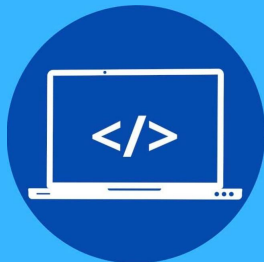
Variable Types

(String, int, double)



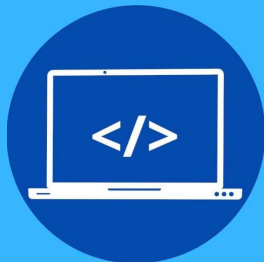
Strings

```
public static void main(String args[])  
{  
    String s = "Hello cruel world";  
    System.out.println(s);  
}
```



ints

```
public static void main(String args[])  
{  
    int age = 69;  
    System.out.println(age);  
}
```



double

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

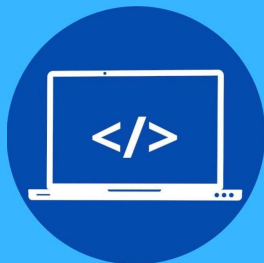
```
{
```

```
double d = -137.8036;
```

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

```
d = 1.45667E23; //Scientific notation...means 1.45667 X 1023
```

```
}
```

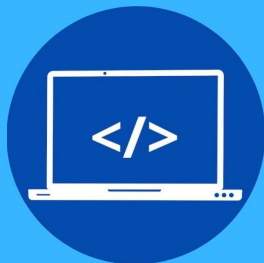


Declaring and initializing

```
double x; //this declares x to be of the double type
```

```
x=1.6; //this initializes x to a value of 1.6
```

```
double x = 1.6 //This both declares and initializes a variable
```

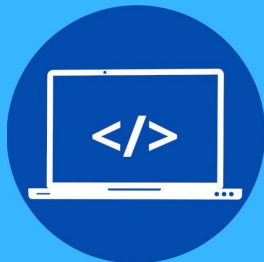


What's legal and what's not:

```
int arws = 47.4;
```

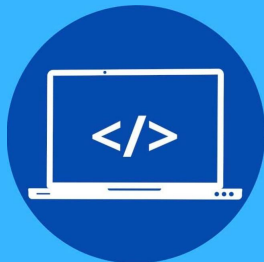
**/illegal, won't compile since a decimal number cannot "fit" into an integer variable.*/*

```
double d = 103; //legal...same as saying the decimal number 103.0
```



Rules for variable names

- Variable names must begin with a letter (or an underscore character) and cannot contain spaces.
- The only “punctuation” character permissible inside the name is the underscore (“_”).
- Variable names cannot be one of the reserved words



Examples

Legal Names

Agro

D

D31

hoppergee

hopper_gee

largeArea

goldNugget

Illegal Names

139

139Abc

Fast One

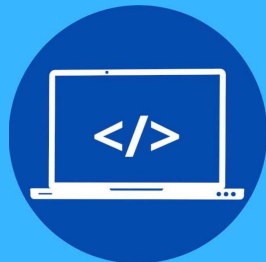
Class

slow.Sally

Double

goldNugget

hopper-gee



Variable Naming Conventions

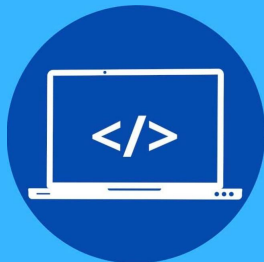
It is traditional for variable names to start with a lowercase letter. If a variable name consists of multiple words, combine them in one of two ways:

`bigValue...` jam everything together. First word begins with a small letter and subsequent words begin with a capital.

`big_value...` separate words with an underscore.



Simple String Operations



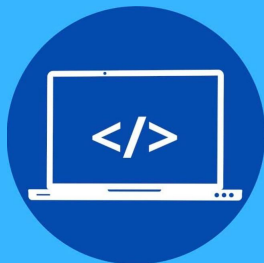
Concatenation

```
String mm = "Hello";
```

```
String nx = "good buddy";
```

```
String c = mm + nx;
```

```
System.out.println(mm + " " + nx); // prints Hello good buddy
```

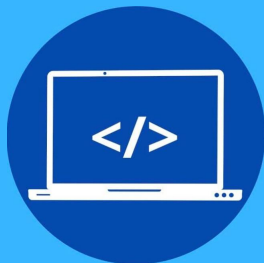


The *length* method

```
String theName = "Donald duck";
```

```
int len = theName.length();
```

```
System.out.println(len); //prints 11
```

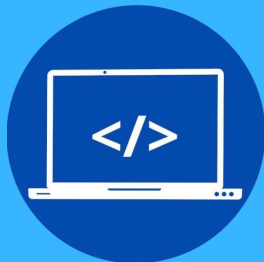


A piece of a *String* (*substring*)

```
String myPet = "Sparky the dog";
```

```
String smallPart = myPet.substring(4);
```

```
System.out.println(smallPart); // prints ky the dog
```



Conversion between lower and upper case

toLowerCase

```
String bismark = "Dude, where's MY car?" ;
```

```
System.out.println(bismark.toLowerCase()); // prints dude, where's my car?
```

toUpperCase

```
System.out.println("Dude, where's My car?" .toUpperCase());
```

```
// prints DUDE, WHERE'S MY CAR?
```



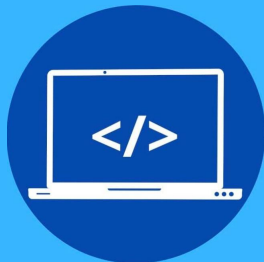
Concatenating a *String* and a numeric

```
int x = 27;
```

```
String x = "Was haben wir gemacht"; // German for "What have we done?"
```

```
String combo = s + " " + x;
```

```
System.out.println(combo); // prints Was haben wir gemacht 27
```



Escape Sequences

- Escape Sequence \"

What "is" the right way?

```
String s = "What \"is\" the right way?";
```

```
System.out.println(s); // prints What "is" the right way?
```

- Escape Sequence \n

```
String s = "Here is one line\nand here is another.";
```

```
System.out.println(s); // prints Here is one line
```

and here is another,



Escape Sequences Cont.

- Escape Sequence \\

```
System.out.println("Path = c:\\nerd_file.doc");
```

Prints the following:

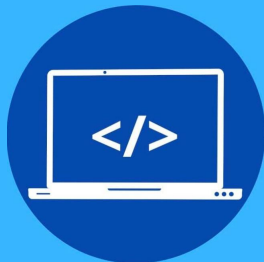
```
Path = c:\nerd_file.doc
```

- Escape Sequence \t

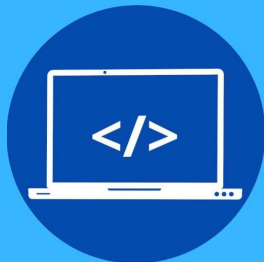
```
System.out.println("Name:\t\tAddress:")
```

Prints the following:

```
Name:           Address:
```



Using Numeric Variables



The Assignment Operator

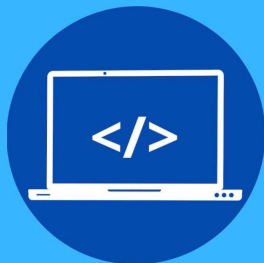
```
int i = 3; // assigning the value 3 to i
```

```
3 = i; // ILLEGAL! Data cannot flow 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

```
double b, mud, puma; // the variables are only declared
```

```
double x = 31.2, M = 37.09, zu, p = 43.917 // x, m , and p declared and  
initialized. zu is just declared
```



Fundamental Arithmetic Operations

The basic arithmetic operations 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.



Fundamental Arithmetic Operations Cont.

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”...



Fundamental Arithmetic Operations Example

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

```
System.out.println(8 - 5 * 6 / 3 + (5 - 6) * 3); //prints -5
```



Not the Same as in Algebra

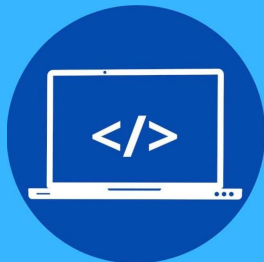
`count = count + 3;` //this is illegal in algebra; however, in computer science it

`//` means the new count equals the old count + 3.

```
int count = 15;
```

```
count = count + 3;
```

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



Increment and Decrement

`x++`; means the same as `x = x + 1`;

`x--`; means the same as `x = x - 1`;

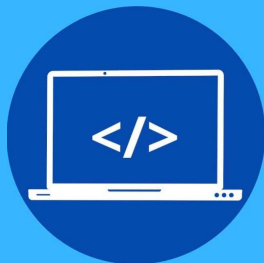
`x++` is the same as `++x` (the `++` can be on either side of `x`)

`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

- +=

>>

x += 3

- -=

>>

x -= y - 2

- *=

>>

z *= 46;

Simplified meaning

x = x + 3;

x = x - (y - 2);

z = z * 46;



Compound Operators Cont.

Syntax Exmple

- /=

>>

p/=x-z;

- %=

>>

j%=2

Simplified Example

p = p / (x-z);

j = j%2;



Code Examples

```
int g = 409;
```

```
g += 5;
```

```
System.out.println(g); // prints 414
```

```
double d = 20.3;
```

```
double m = 10.0;
```

```
m *= d - 1;
```

```
System.out.println(m); // prints 193.0
```



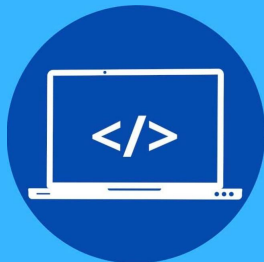
The Whole Truth

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

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

`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;
```

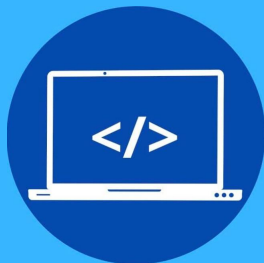
```
int p = 2 + q++;
```

```
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

```
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
```

