Class

A class is a user defined blueprint or prototype from which objects are created.
It represents the set of properties or methods that are common to all objects of one type

Class -> Actions / Verifications (Method)

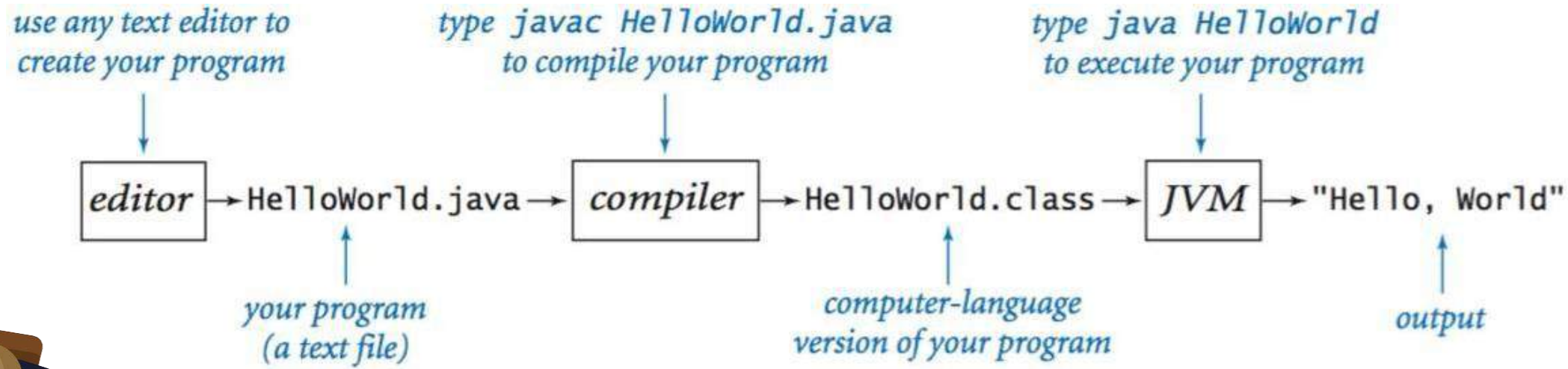
driveCar
applyBreak
soundHorn
isPuncture

The name must not contain any white spaces.
The name should not start with special characters like & (ampersand), $ (dollar), _ (underscore).

Class -> Information / Data (Variable)

bodyColor
numWheels
registrationNumber

How Java works?



Primitive Data Types

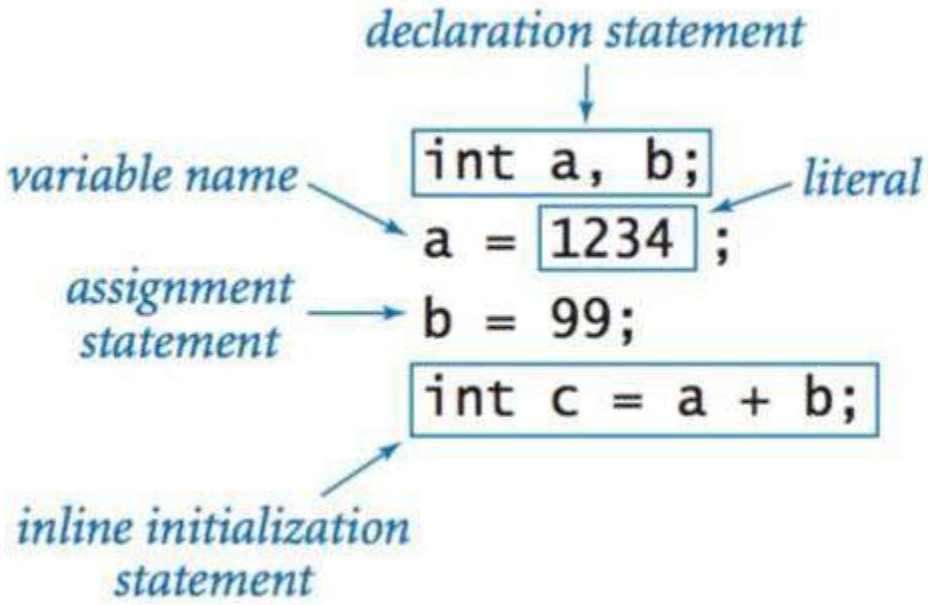
Type	Description	Default	Size	Example Literals
boolean	true or false	false	1 bit	true, false
byte	twos complement integer	0	8 bits	(none)
char	Unicode character	\u0000	16 bits	'a', '\u0041', '\101', '\\', '\'', '\n', 'ß'
short	twos complement integer	0	16 bits	(none)
int	twos complement integer	0	32 bits	-2, -1, 0, 1, 2
long	twos complement integer	0	64 bits	-2L, -1L, 0L, 1L, 2L
float	IEEE 754 floating point	0.0	32 bits	1.23e100f, -1.23e-100f, .3f, 3.14F
double	IEEE 754 floating point	0.0	64 bits	1.23456e300d, -1.23456e-300d, 1e1d

Class & Main method

```
public class MyFirstApp {

    public static void main (String[] args) {

        System.out.print("I Rule!");

    }

}
```

- public so everyone can access it
- this is a class (duh)
- the name of this class
- opening curly brace of the class
- (we'll cover this one later.)
- the return type. void means there's no return value.
- the name of this method
- arguments to the method. This method must be given an array of Strings, and the array will be called 'args'
- opening brace of the method
- this says print to standard output (defaults to command-line)
- the String you want to print
- every statement MUST end in a semicolon!!
- closing brace of the main method
- closing brace of the MyFirstApp class

Declaration and Assignment



Comparison Operators

op	meaning	true	false
==	equal	2 == 2	2 == 3
!=	not equal	3 != 2	2 != 2
<	less than	2 < 13	2 < 2
<=	less than or equal	2 <= 2	3 <= 2
>	greater than	13 > 2	2 > 13
>=	greater than or equal	3 >= 2	2 >= 3

Conditional Statement

```
boolean expression
        ↓
if ( x > y )
{
    int t = x;
    x = y;
    y = t;
}
```

sequence of statements →

Conditional statement:

if statement

syntax;

```
if(condition){
----------
}
```

if else statement

syntax;

```
if(condition){
----------
}
else{
----
}
```

If else

absolute value

```
if (x < 0) x = -x;
```

put the smaller value in x and the larger value in y

```
if (x > y)
{
   int t = x;
   x = y;
   y = t;
}
```

maximum of x and y

```
if (x > y) max = x;
else       max = y;
```

error check for division operation

```
if (den == 0) System.out.println("Division by zero");
else          System.out.println("Quotient = " + num/den);
```

error check for quadratic formula

```
double discriminant = b*b - 4.0*c;
if (discriminant < 0.0)
{
   System.out.println("No real roots");
}
else
{
   System.out.println((-b + Math.sqrt(discriminant))/2.0);
   System.out.println((-b - Math.sqrt(discriminant))/2.0);
}
```

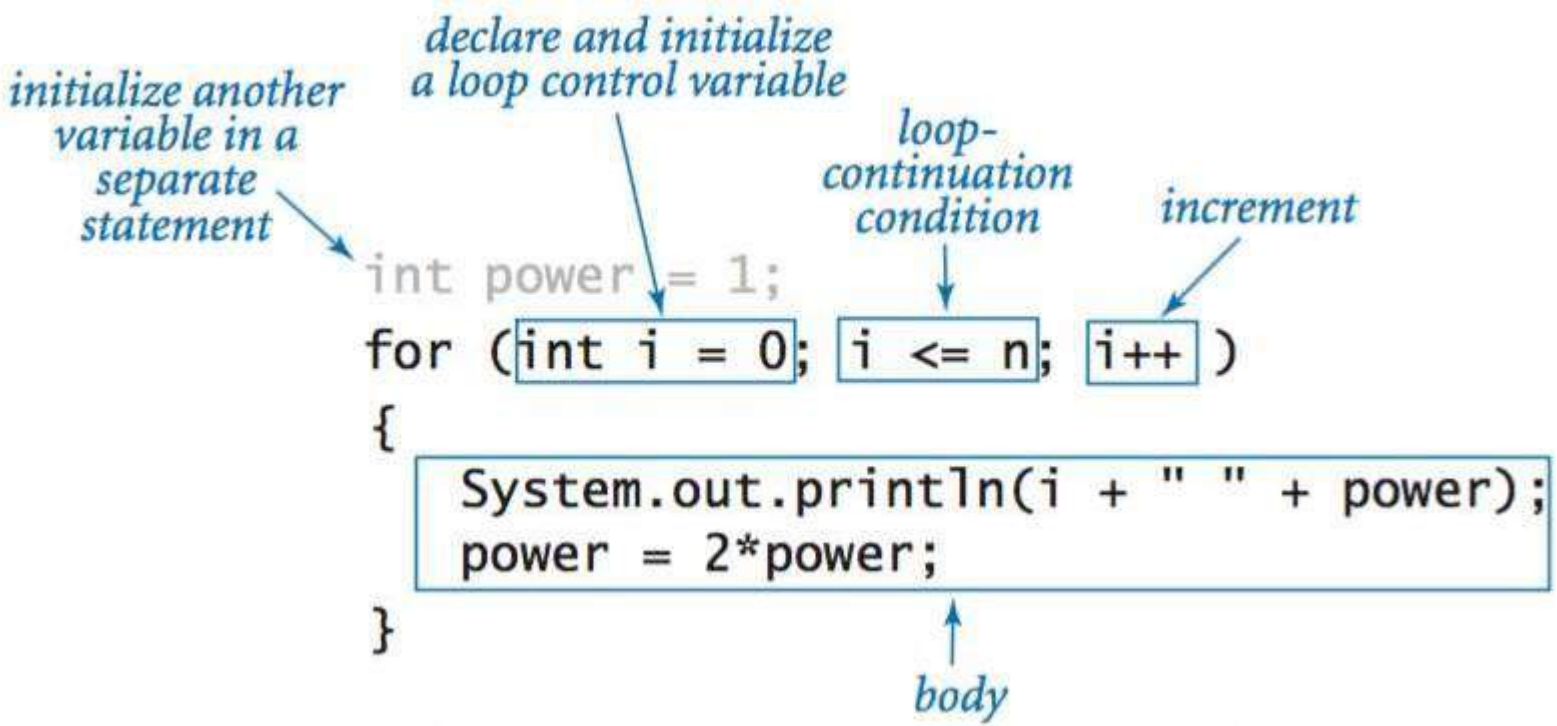
if else if (or) nested if statement

syntax;
if(condition){
- - - - - - - - - - -
}
else if (condition){
- - - -
}
else{
- - - - - - - -
}
Looping Statement:

For loop

syntax:
for(initialization;
condition;increment/decrement) for(int
i=0;i<10;i++){
- - - - - -
}
1st initialization
2nd check condition
3rd execution ,4 increment

For loop.


initialize another variable in a separate statement
declare and initialize a loop control variable
loop-continuation condition
increment
int power = 1;
for (int i = 0; i <= n; i++)
{
System.out.println(i + " " + power);
power = 2*power;
}
body

Break statement.

Break Statement is a loop control statement that is used to terminate the loop. As soon as the break statement is encountered from within a loop, the loop iterations stop there, and control returns from the loop immediately to the first statement after the loop.

```
for (int i = 1; i <=5; i++) {
if(i==3) {
System.out.println("Three"
); break;
}


}System.out.println("Loop Complete");
```

Output:

Three
Loop Complete

Continue statement.

when a continue statement is encountered the control directly jumps to the beginning of the loop for the next iteration instead of executing the statements of the current iteration. The continue statement is used when we want to skip a particular condition and continue the rest execution.

```
for (int i = 1; i <=5; i++) {
if(i==3) {
System.out.println("Three"
); continue;
}

System.out.println(i);
}
```

Output:

1
2
Three
4
5

Method Signature