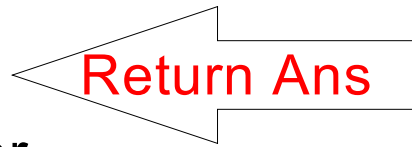


Methods ... arguments & parameters

Consider these scenarios ...

METHOD

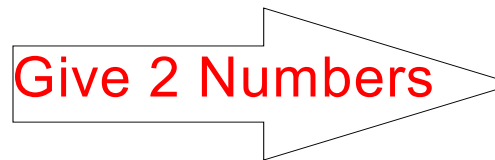
Ask a method to give back to you a value which is triple the value it will ask the user to enter



Ask user for a number & triple value (Ans)

A RETURN VALUE is provided by the method

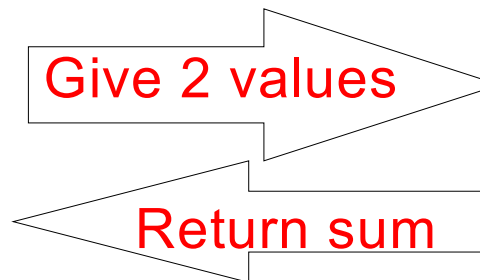
Send a method two different numbers, and have the method print out the difference



Calculate and display the difference between the two numbers

ARGUMENTS are PASSED into the PARAMETERS of the method.

Send two values to a method, have it calculate the sum, and return that sum back for the program to continue working



Calculate sum of the two numbers received, but return the answer back to the program

ARGUMENTS are PASSED to the PARAMETERS, and a RETURN VALUE is generated.

Methods ... arguments & parameters

CALLING
the
FUNCTIONAL
METHOD

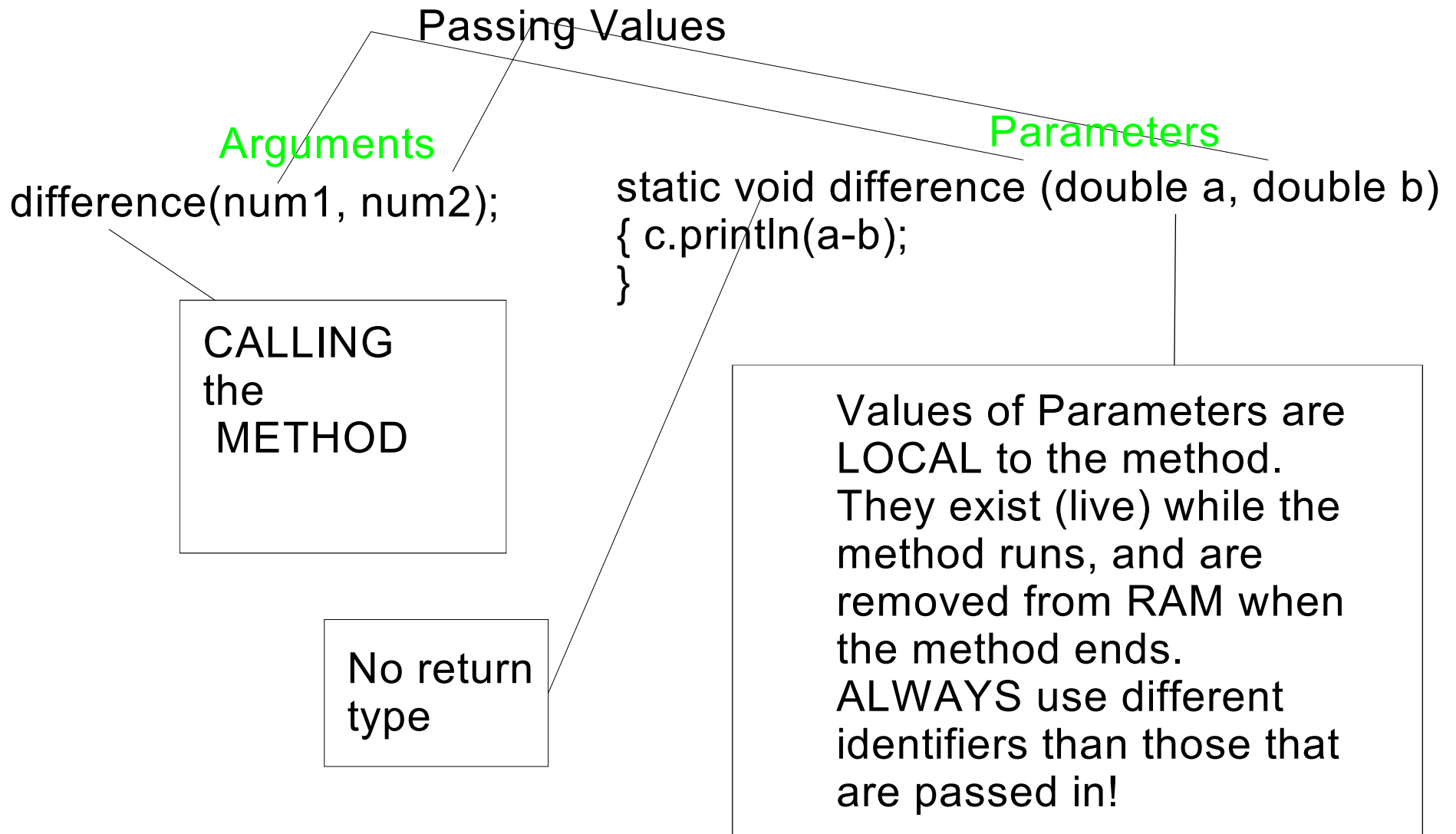
tripple_num = **triple()**;

RETURN TYPE, which
must be the same as the
value it returns to
(ie tripple_num is int)

```
static int triple()  
{ int num;  
  c.print ( "Enter number" );  
  num = c.readInt();  
  num = num * 3;  
  return num;  
}
```

Sending a value back to
the identifier that stores
the method's result
(tripple_num)

Methods ... arguments & parameters



Methods ... arguments & parameters

Pass in arguments to
the parameters of the
method (same TYPE)

total = sum(num1, num2);

```
static int sum( int a, int b)  
{ return a + b; }
```

Return value to where
method was called.
(Same TYPE ... int in this
case)

Methods ... arguments & parameters

Definitions:

method	- a group of code which performs a specific task
procedural method	- a method which executes common code
functional method	- a method which returns a value
arguments	- values passed into a method, transferred to the parameter values
parameters	- the local identifiers in a method which receive values from the 'call'
call	- 'invoking' the method, the act of making a method run
return value	- an identifier which is passed FROM a method into an identifier assigned to the method call
void	- indicates a procedural method ... no return type

Methods ... arguments & parameters

Rules:

- Arguments and parameters MUST match in both TYPE or ORDER of presentation
- Arguments and parameters can be of mixed TYPE, as long as the order is the same
- Return values MUST match the TYPE of identifier it is being returned to
- An entire array can be passed into and returned from methods by simply using the array name ... the name without the [subscript] is an OBJECT which holds a POINTER to the location of the array
- Methods ALWAYS have descriptive names, and are found outside the { } of the 'main' method, but within the class set of of { }. Normally, methods occur after the main control method
- Methods can 'call' methods. They always return from the call point.