Walkthrough of a Method Call

Michael Clarkson

This is the code this example will use:

```java
class MethodCall {
    static int combine(int x, int y) {
        x = y - x;
        y = x + y;
        System.out.println(x);
        System.out.println(y);
        return x;
    }
    public static void main(String[] args) {
        int a = 1, b = 3;
        System.out.println(a);
        System.out.println(b);
        a = combine(a, b);
        System.out.println(a);
        System.out.println(b);
    }
}
```

This is where we’ll keep track of the values of variables that are in scope:

This arrow will show where we are in the code:

This is where we’ll record the output from the program:

The first statement declares and initializes two variables:

The next two statements print the values of those variables:

Execution always starts with the method called main:
Then we call a method.

```java
class MethodCall {
    static int combine(int x, int y) {
        x = y - x;
        y = x + y;
        System.out.println(x);
        System.out.println(y);
        return x;
    }
    public static void main(String[] args) {
        int a = 1, b = 3;
        System.out.println(a);
        System.out.println(b);
        a = combine(a, b);
        System.out.println(a);
        System.out.println(b);
    }
}
```

When we start executing the method, the parameters are initialized to the arguments.

Then the second assignment...

```java
class MethodCall {
    static int combine(int x, int y) {
        x = y - x;
        y = x + y;
        System.out.println(x);
        System.out.println(y);
        return x;
    }
    public static void main(String[] args) {
        int a = 2, b = 3;
        System.out.println(a);
        System.out.println(b);
        a = combine(a, b);
        System.out.println(a);
        System.out.println(b);
    }
}
```

Now we return to where we were called from:

```java
class MethodCall {
    static int combine(int x, int y) {
        x = y - x;
        y = x + y;
        System.out.println(x);
        System.out.println(y);
        return x;
    }
    public static void main(String[] args) {
        int a = 2, b = 3;
        System.out.println(a);
        System.out.println(b);
        a = combine(a, b);
        System.out.println(a);
        System.out.println(b);
    }
}
```
And the return value gets substituted for the method call:

class MethodCall {
    static int combine(int x, int y) {
        x = y - x;
        y = x + y;
        System.out.println(x);
        System.out.println(y);
        return x;
    }
    public static void main(String[] args) {
        int a = 1, b = 3;
        System.out.println(a);
        System.out.println(b);
        a = combine(a, b);
        System.out.println(a);
        System.out.println(b);
    }
}

Finally, we print out the variables again:

class MethodCall {
    static int combine(int x, int y) {
        x = y - x;
        y = x + y;
        System.out.println(x);
        System.out.println(y);
        return x;
    }
    public static void main(String[] args) {
        int a = 1, b = 3;
        System.out.println(a);
        System.out.println(b);
        a = combine(a, b);
        System.out.println(a);
        System.out.println(b);
    }
}