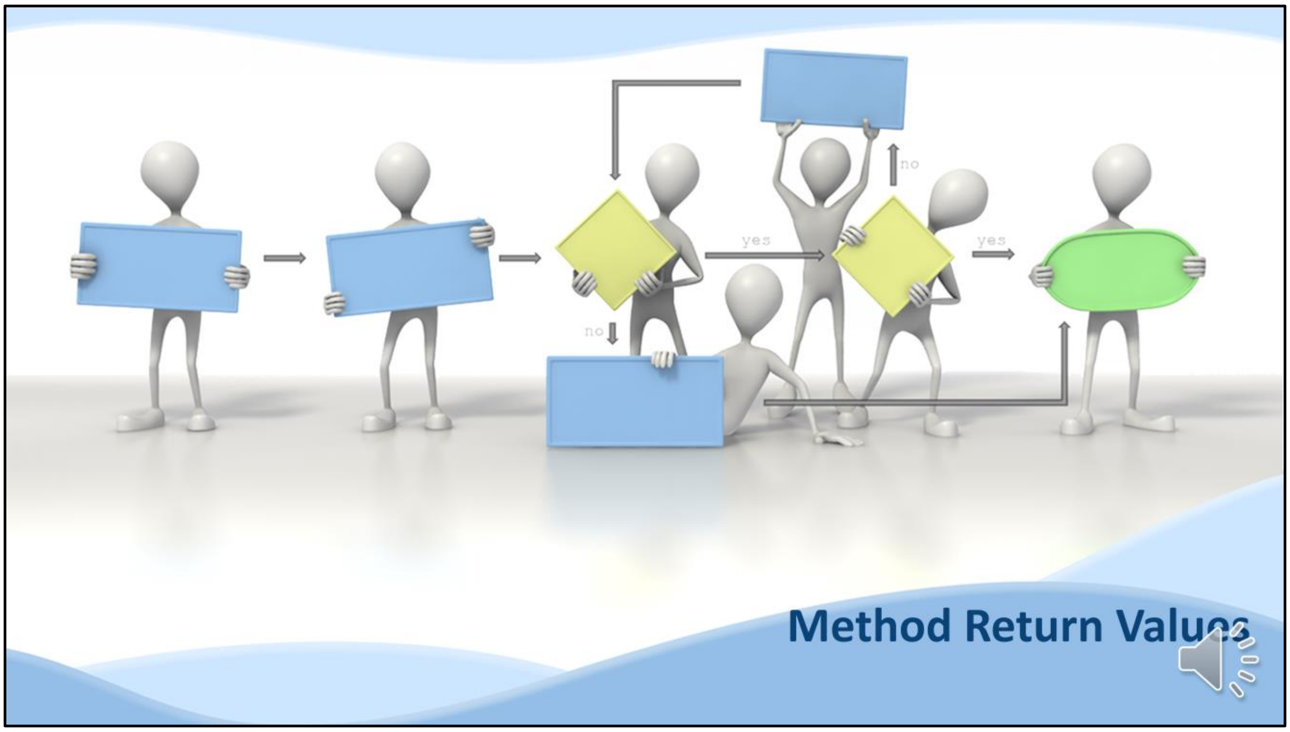
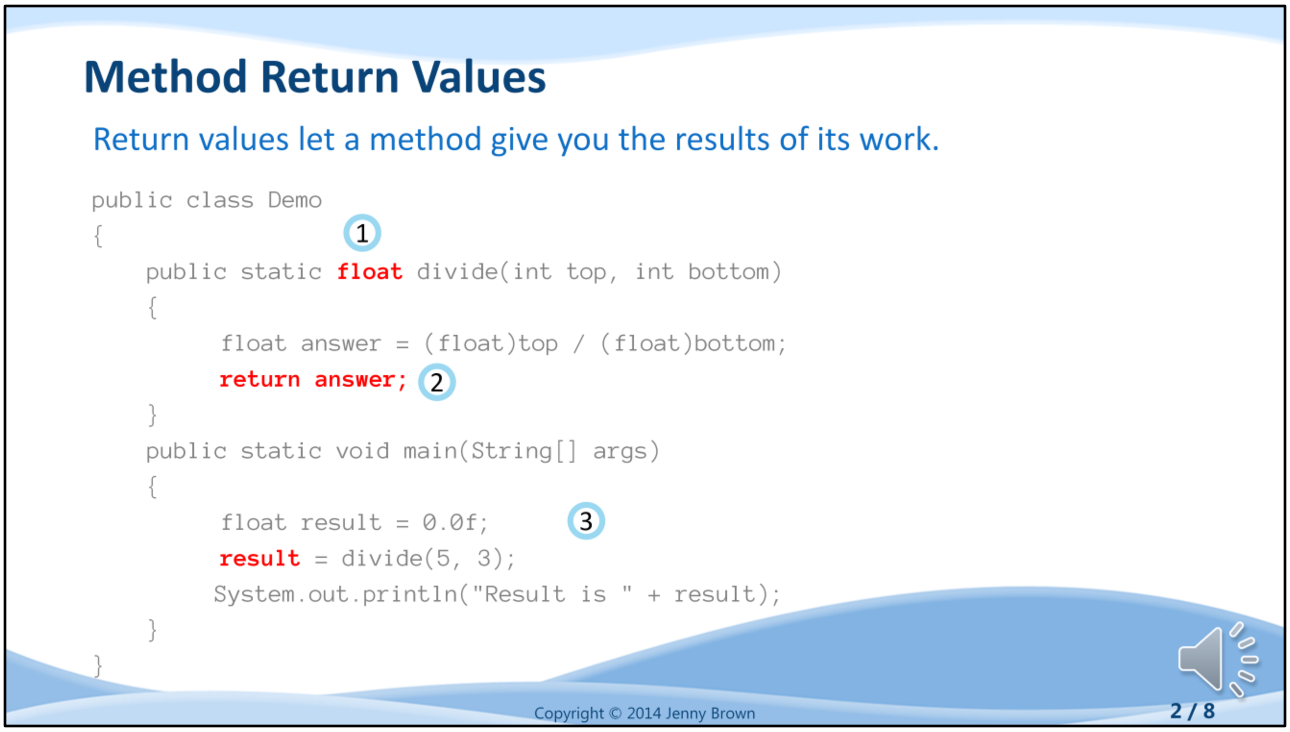


Method Return Values

A method can optionally have a return value, which lets it give you back the results of some work it has done. Let's take a look.

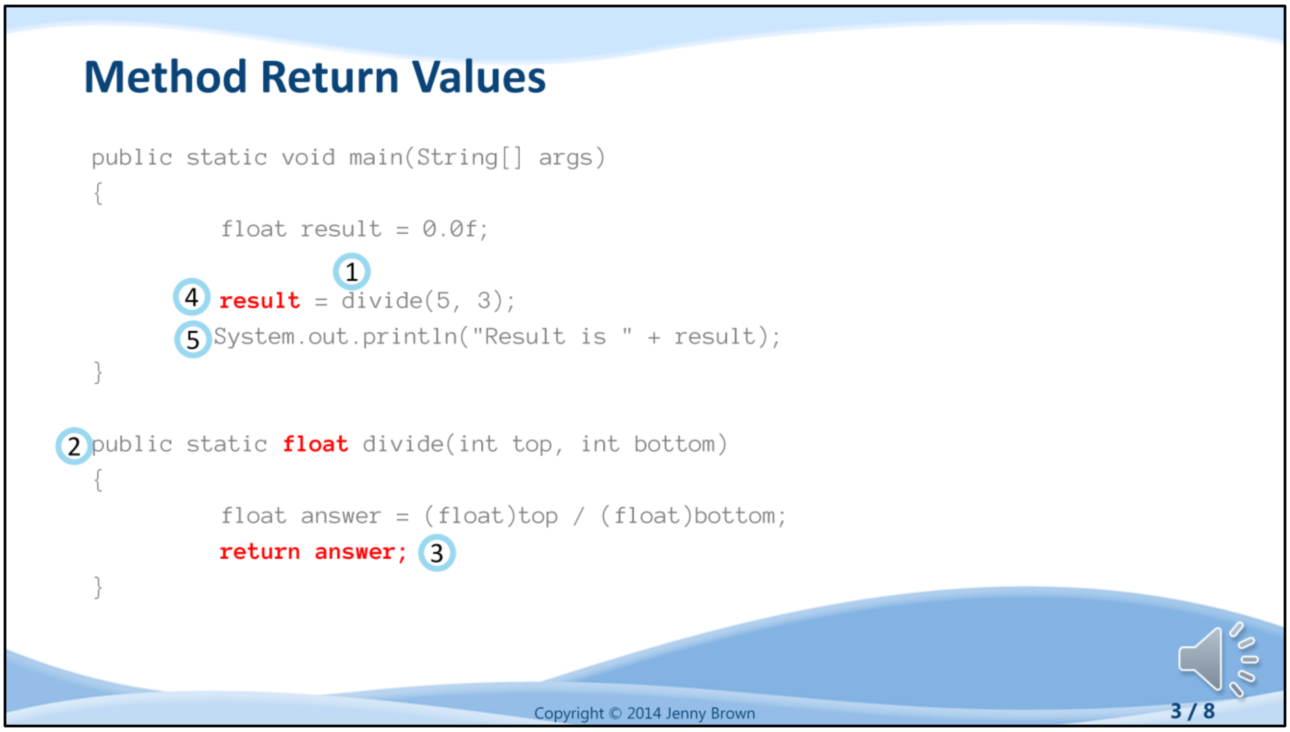


Method Return Values

Return values let a method give you the results of its work.

```
public class Demo
{
    public static float divide(int top, int bottom)
    {
         float answer = (float)top / (float)bottom;
         return answer;
    }
    public static void main(String[] args)
    {
         float result = 0.0f;
         result = divide(5, 3);
        System.out.println("Result is " + result);
    }
}
```

This example has a divide() method that takes two numbers, calculates the first divided by the second, and gives an answer back. Notice at (1) that I am no longer using the "void" modifier. Instead, I'm saying "float". Why? That modifier indicates the return type of the method. In this case, the divide method returns a float - specifically, it returns "answer", noted at location (2). When I make a method call at location (3), I don't just call the method - I also assign a variable "result" with the answer that divide() gives us back.



Method Return Values

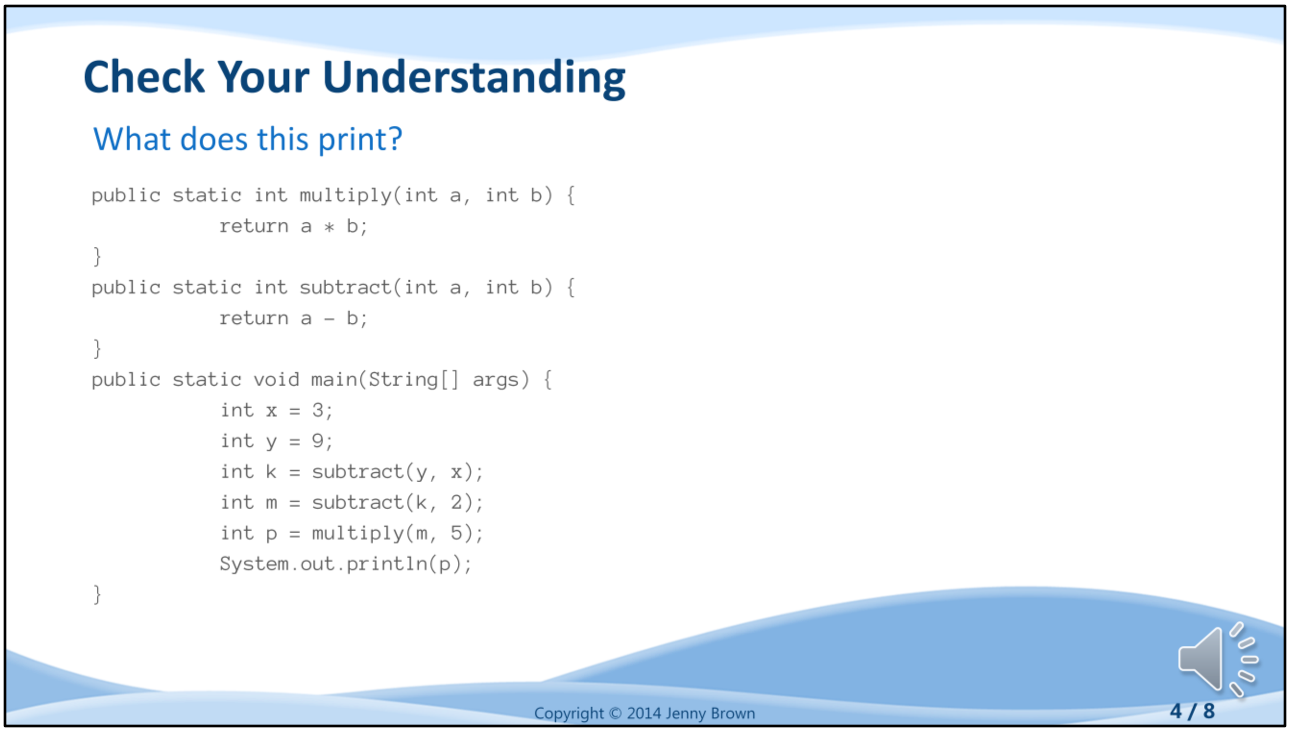
```
public static void main(String[] args)
{
         float result = 0.0f;
                          ①
      ④ result = divide(5, 3);
      ⑤ System.out.println("Result is " + result);
}

②public static float divide(int top, int bottom)
{
         float answer = (float)top / (float)bottom;
         return answer; ③
}
```

Let's look at the flow through the code. I've rearranged this a little to make it easier to describe. Let's start with main() at the top. We declare a variable named result and give it an initial value of zero. Then we want to assign it the value that comes back from divide, but we don't have the value yet. So the computer pauses in the middle of that line and jumps to the other part of the code, to run divide.

This jumps us from location (1) to location (2). The method input parameter named top gets the value 5, since that was the first integer, and the parameter named bottom gets 3, from the second integer. Then we calculate answer as 5 divided by 3. Then we use a special keyword, "return" to send that answer back as the result of calling divide. That makes the jump from location (3) to location (4).

At location (4), the return value from divide gets assigned to the variable result. Now we've completed running that line. Next, at location (5), we print a message to the screen that includes the calculated result.



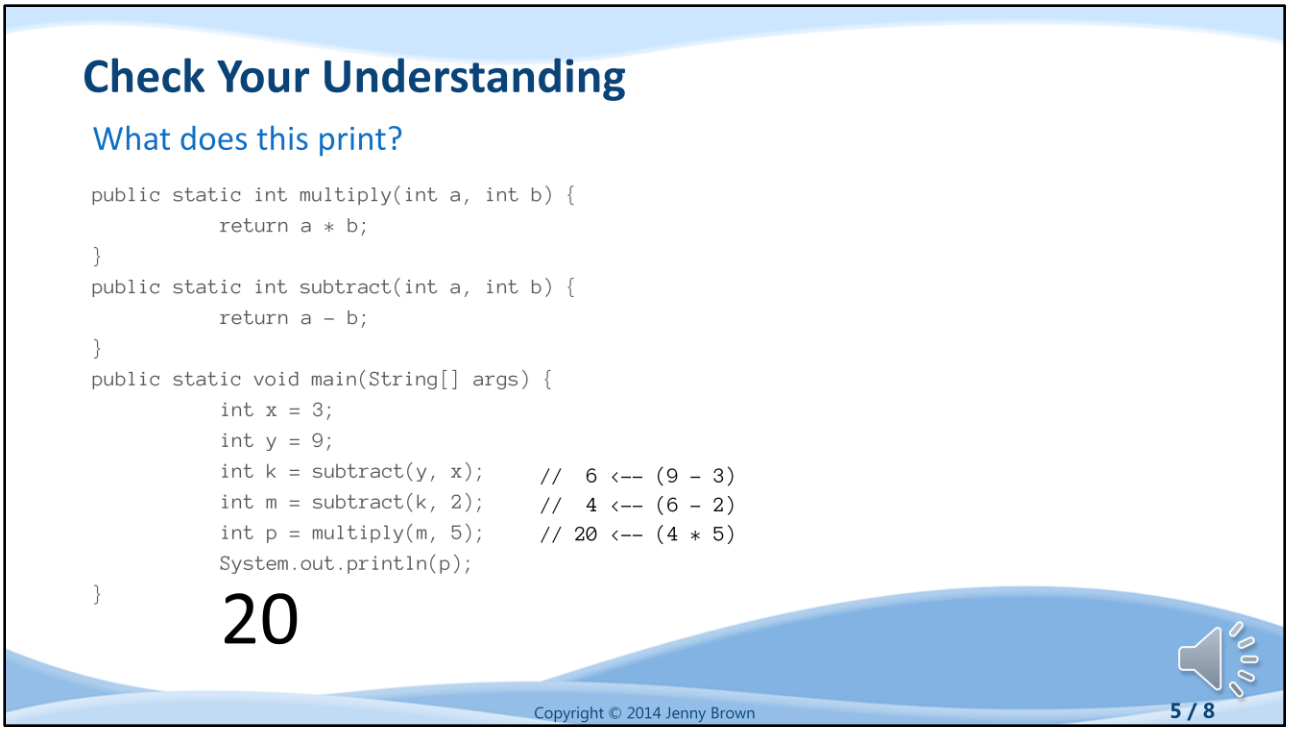
Check Your Understanding

What does this print?

```
public static int multiply(int a, int b) {
        return a * b;
}
public static int subtract(int a, int b) {
        return a - b;
}
public static void main(String[] args) {
        int x = 3;
        int y = 9;
        int k = subtract(y, x);
        int m = subtract(k, 2);
        int p = multiply(m, 5);
        System.out.println(p);
}
```

Here's an example to check your understanding. Grab scratch paper and work through this example with the video paused. The next slide will show you the answer so you can check your work.

Take your time and think this through.



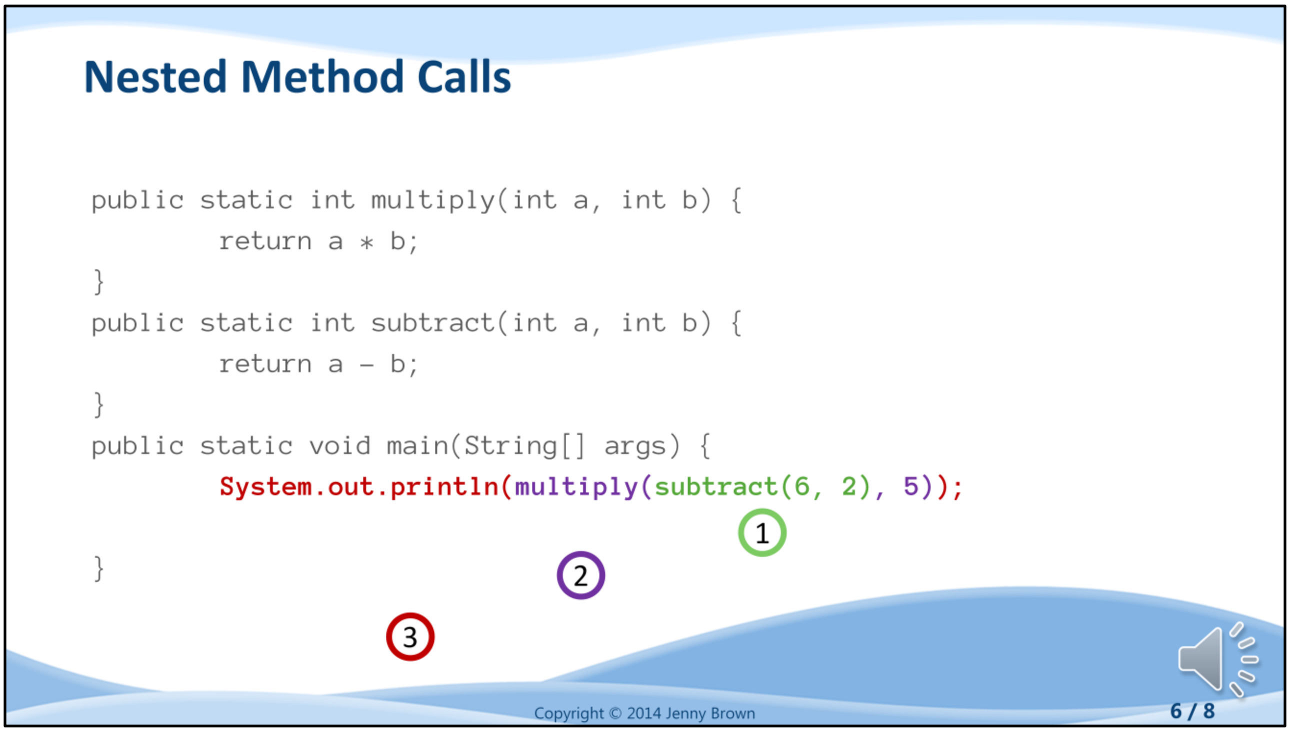
Check Your Understanding

What does this print?

```
public static int multiply(int a, int b) {
          return a * b;
}
public static int subtract(int a, int b) {
          return a - b;
}
public static void main(String[] args) {
          int x = 3;
          int y = 9;
          int k = subtract(y, x);     //  6 <-- (9 - 3)
          int m = subtract(k, 2);     //  4 <-- (6 - 2)
          int p = multiply(m, 5);     // 20 <-- (4 * 5)
          System.out.println(p);
}
```

20

So the answer it prints out is twenty. The comments explain how the math works at each method call.



Nested Method Calls

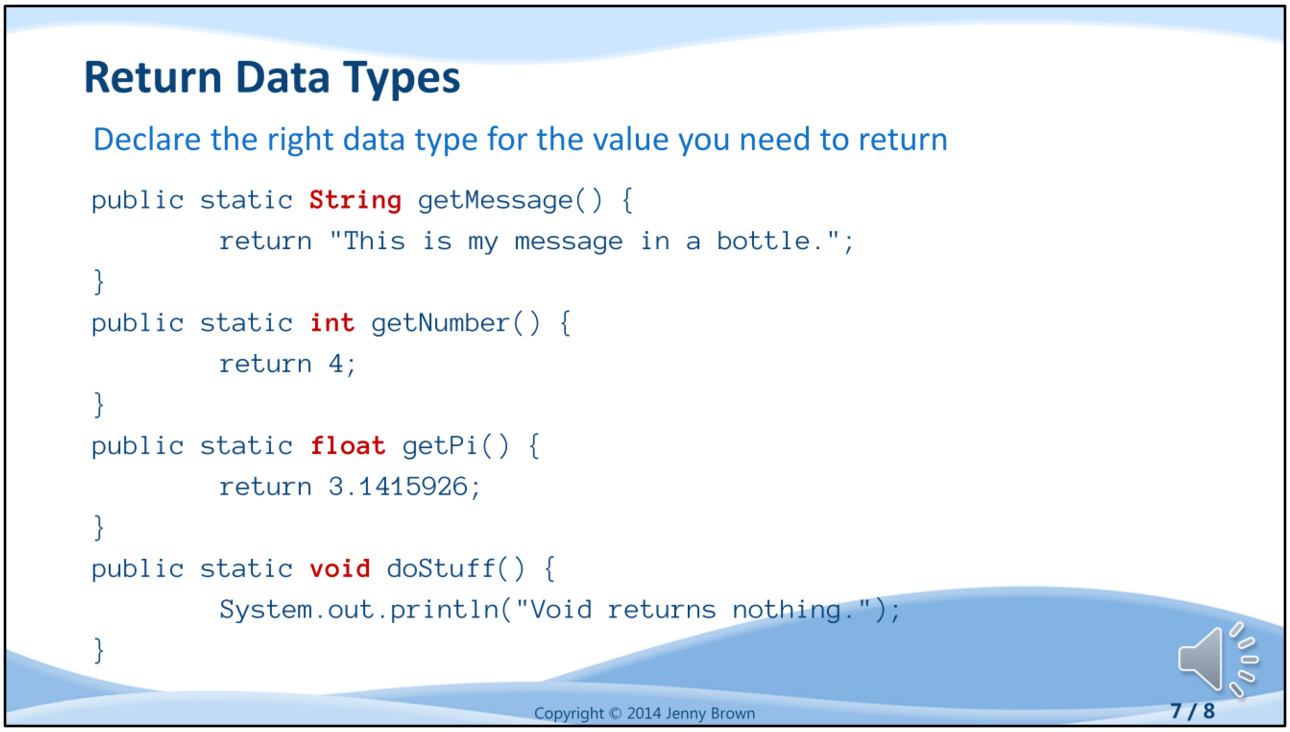
```
public static int multiply(int a, int b) {
        return a * b;
}
public static int subtract(int a, int b) {
        return a - b;
}
public static void main(String[] args) {
        System.out.println(multiply(subtract(6, 2), 5));

}
```

6 / 8

Here's an example of nested method calls, one inside another.

In main(), the computer first runs the subtract() method call, and calculates 6 minus 2. That produces a return value of 4. Then it puts the return value in place of the method call, so subtract(6, 2) becomes 4. That makes the multiply() call become multiply(4, 5). Then multiply runs and calculates 4 times 5 which produces a return value of 20. The value 20 is returned and takes the place of the entire multiply() call. That makes the outermost call be System.out.println(20); so the computer prints 20 to screen.



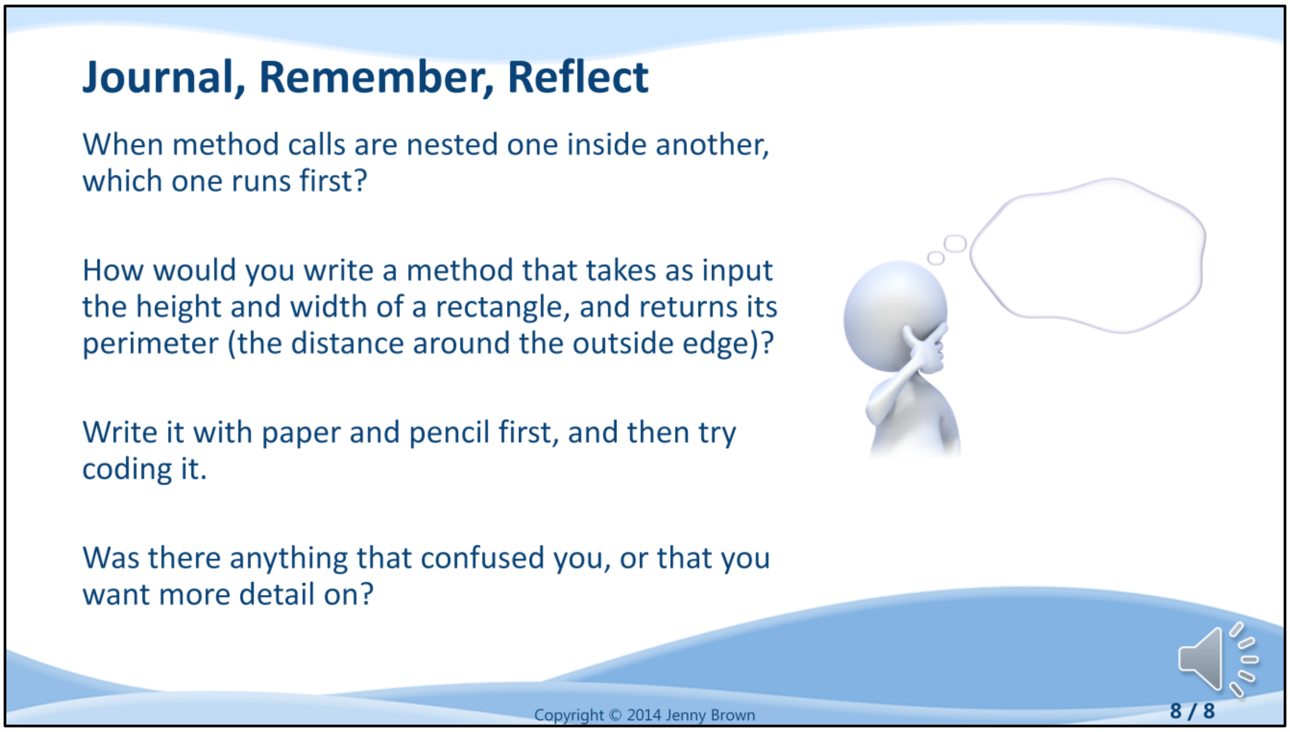
Return Data Types

Declare the right data type for the value you need to return

```
public static String getMessage() {
        return "This is my message in a bottle.";
}
public static int getNumber() {
        return 4;
}
public static float getPi() {
        return 3.1415926;
}
public static void doStuff() {
        System.out.println("Void returns nothing.");
}
```

When you're planning methods with return values, you must declare the return type to properly match what value you're returning. In this first example, the message is a string of words, so we declare type String. In the second example, the return value is an integer, so we declare type int. And in the third example, the return value is a floating point number, so we declare float. The last example shows what to use when there is no return value. When a method does not need a return value, the return type is void.

The compiler will give you an error if the value's type is different from the method declaration's return type. You can use any data type, including boolean and others that aren't listed here, but it must match what is specified in the method definition.



Journal, Remember, Reflect

When method calls are nested one inside another, which one runs first?

How would you write a method that takes as input the height and width of a rectangle, and returns its perimeter (the distance around the outside edge)?

Write it with paper and pencil first, and then try coding it.

Was there anything that confused you, or that you want more detail on?

8 / 8

Journal time.

When method calls are nested one inside another, which one runs first?

How would you write a method that takes as input the height and width of a rectangle, and returns its perimeter (the distance around the outside edge)?

Write it with paper and pencil first, and then try coding it.

Was there anything that confused you, or that you want more detail on?