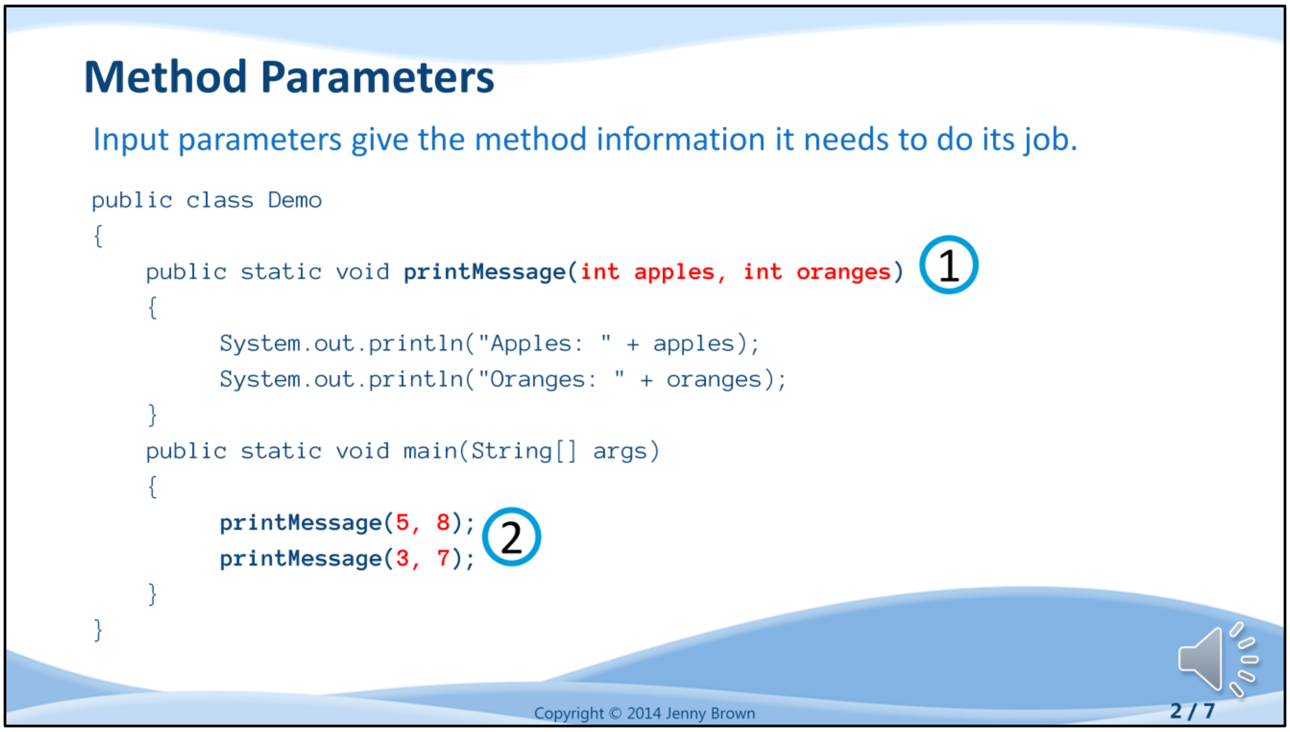
Method Parameters

yes
yes
no
no

What else can methods do? Well, it would help if we could give them extra information to customize the results. How can we do that?



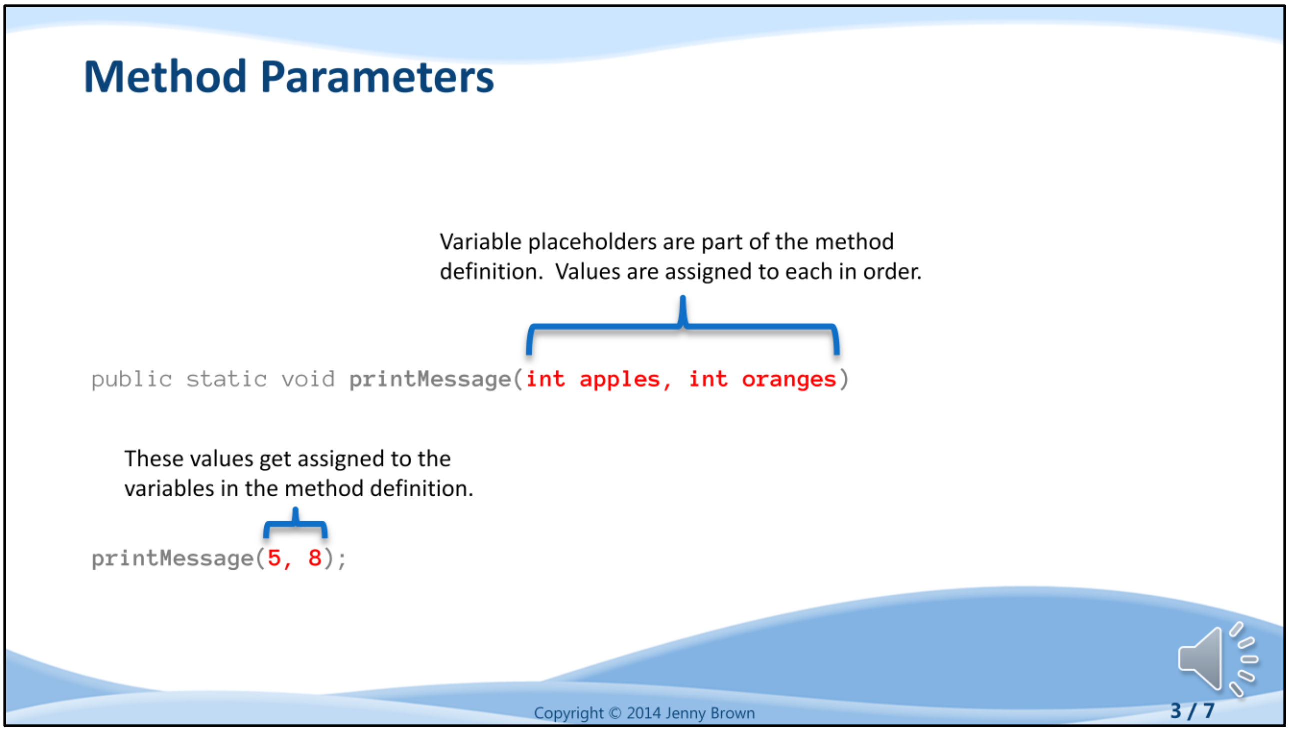
Method Parameters

Input parameters give the method information it needs to do its job.

```
public class Demo
{
    public static void printMessage(int apples, int oranges)   ①
    {
         System.out.println("Apples: " + apples);
         System.out.println("Oranges: " + oranges);
    }
    public static void main(String[] args)
    {
         printMessage(5, 8);   ②
         printMessage(3, 7);
    }
}
```

This example demonstrates defining a method printMessage() and giving it some input parameters as part of that definition. When we create the printMessage() method <click>, we declare two variables inside the parentheses. Each of these are of type integer. That makes their names available within this method, but only there. Now, how do they get their values?

Look in the main() method. <click> This is where we call the method and pass in parameter values for each of the variables. We can even call it multiple times in a row with different values.



Method Parameters

Variable placeholders are part of the method definition. Values are assigned to each in order.

```
public static void printMessage(int apples, int oranges)
```

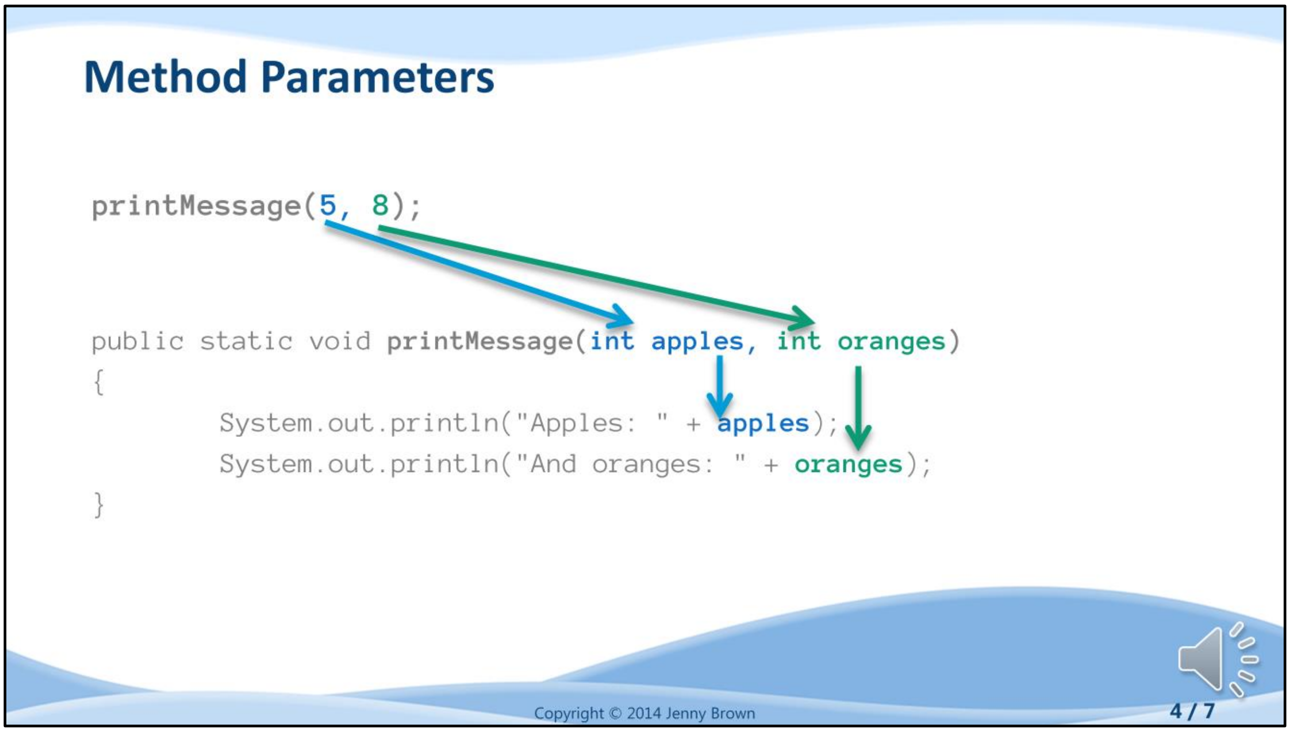
These values get assigned to the variables in the method definition.

```
printMessage(5, 8);
```

3 / 7

Let's take a closer look at the pieces. The variable placeholders are a part of the method definition. They give a data type and a variable name to each parameter. Then, when you make the method call, you must pass in values to match each placeholder. If the number of values is different - for instance, if you passed in a number for apples but nothing for oranges - then you would get a compile error. Try it - what message do you get?

The number and types of parameters must exactly match between the method call and the method definition, and they have to be provided in the same order.



Method Parameters

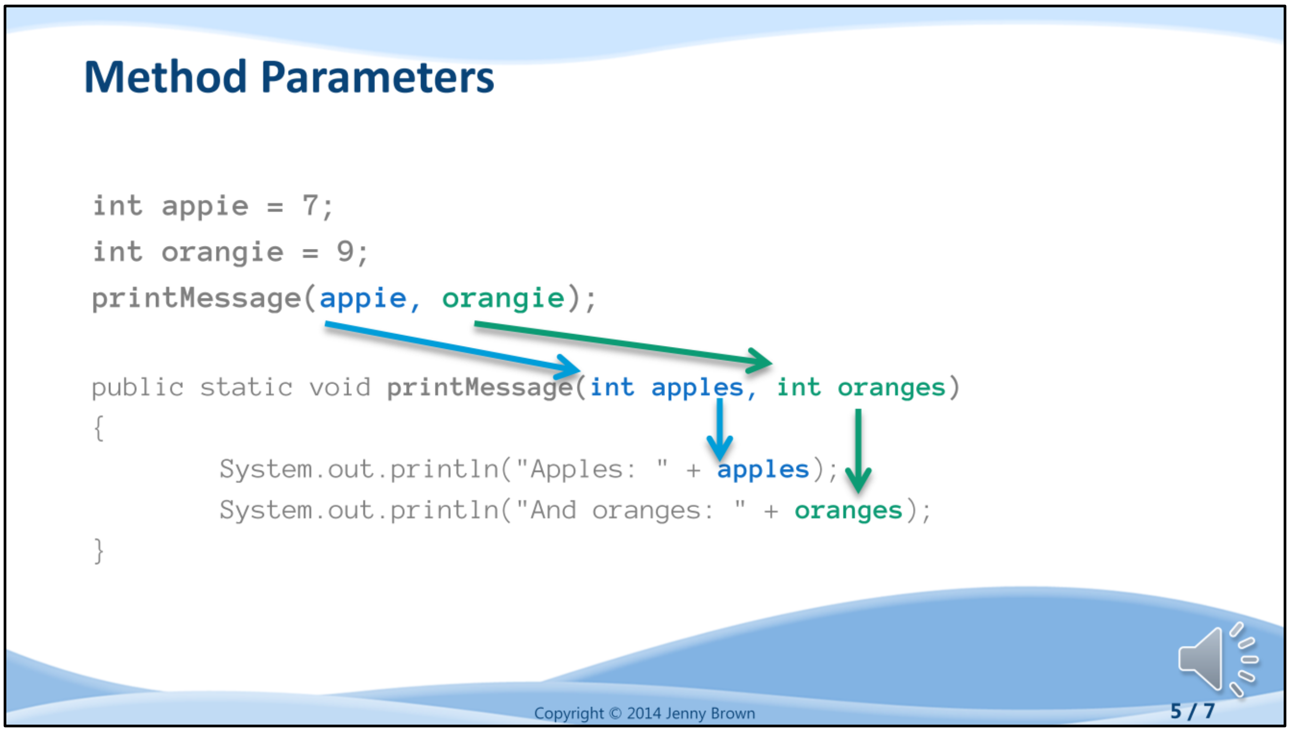
```
printMessage(5, 8);


public static void printMessage(int apples, int oranges)
{
        System.out.println("Apples: " + apples);
        System.out.println("And oranges: " + oranges);
}
```

4 / 7

Here's a visual example that helps me think about how the data moves. I've put the method call at the top just to make the arrows easier to draw.

First, I call printMessage, and give it parameters 5 and 8. The system looks for a method named printMessage, which takes exactly two integer parameters. Then it calls that method, assigning the value 5 to the first variable "apples", and assigning 8 to the second variable, "oranges." It doesn't care what they are named. It only cares what order they appear in.

Once the system is running code inside the method, the variable parameters "apples" and "oranges" exist and have 5 and 8 in them. So now you can use them in your code. This is somewhat similar to declaring these variables inside the method and assigning them values directly. But, doing it in the method declaration, like shown here, means they can get fresh values assigned each time the method is called.



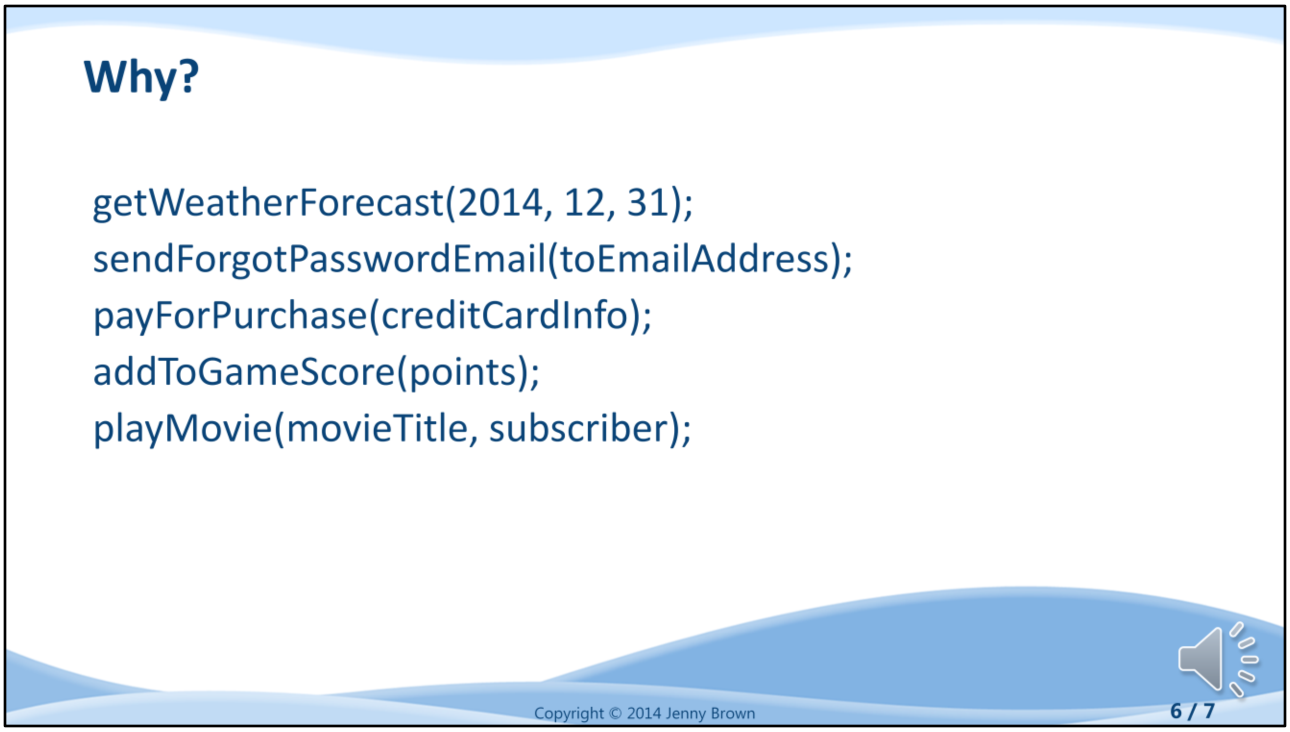
Method Parameters

```
int appie = 7;
int orangie = 9;
printMessage(appie, orangie);

public static void printMessage(int apples, int oranges)
{
        System.out.println("Apples: " + apples);
        System.out.println("And oranges: " + oranges);
}
```

5 / 7

This example shows that you can use variables when calling a method, instead of the actual numbers. It still works the same way. The values inside "appie" and "orangie", which are 7 and 9, are passed into the parameters named apples and oranges. Once inside the method, you use the "apples" and "oranges" names, not the prior "appie" and "orangie" names.

Usually you will choose variable names that are meaningful, so the names in the method definition and the method call might be similar, but they are not required to be the same. The order is important. The data types are important. The names are not.



Why?

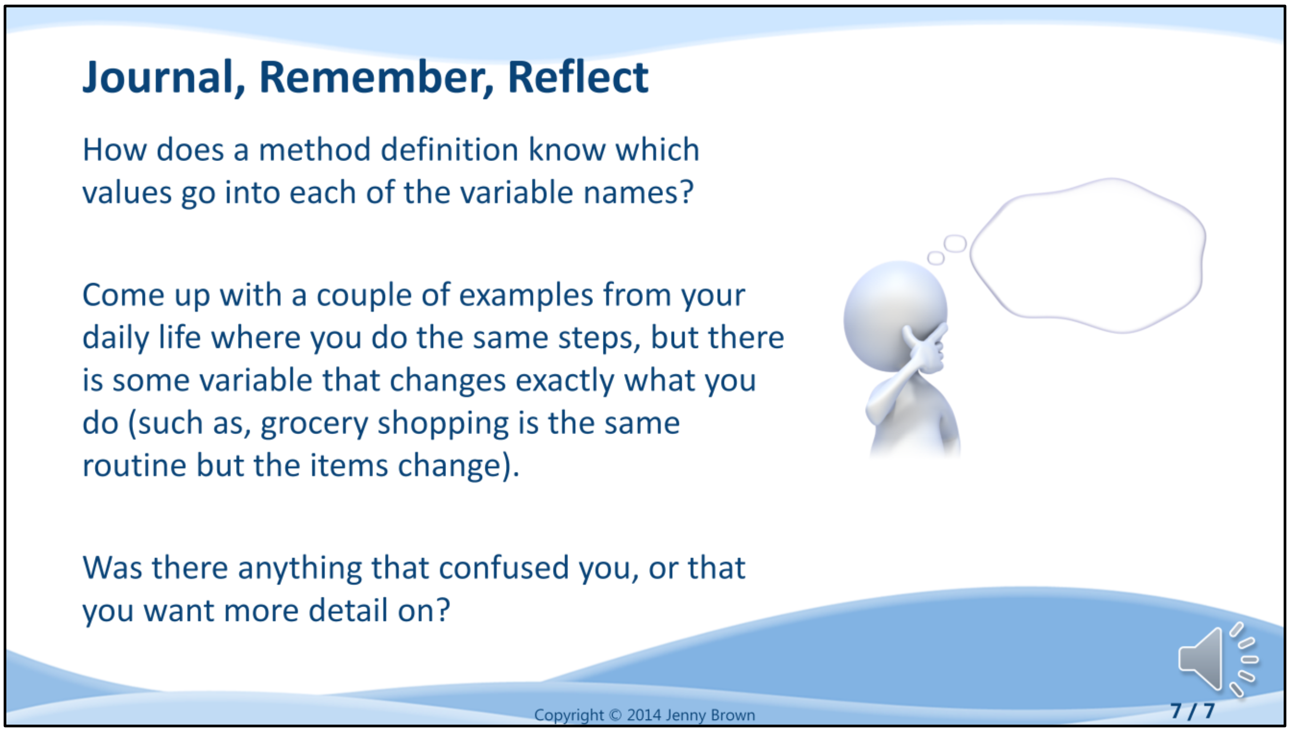
```
getWeatherForecast(2014, 12, 31);
sendForgotPasswordEmail(toEmailAddress);
payForPurchase(creditCardInfo);
addToGameScore(points);
playMovie(movieTitle, subscriber);
```

6 / 7

So why do we use methods? Here are some practical examples. I'm focusing just on the method call, not the method definition, to keep the examples shorter.

Methods let us isolate a section of code so it can be reused. For instance, getting the weather forecast can be done the same way no matter which date we're curious about. Sending an email password reset is the same no matter whose account it is; we just need to know what email address to send the reset to. Paying for an online purchase is the same no matter which credit card it's charged to. And adding to a game score is the same whether we add 1 point, 3 points, or a hundred points. For playing a streaming movie, we just need to know which movie.

All of these are examples of complex code that can be given a name and then reused. The upcoming assignments won't ask you to make anything this complicated, but the idea is the same.



Journal, Remember, Reflect

How does a method definition know which values go into each of the variable names?

Come up with a couple of examples from your daily life where you do the same steps, but there is some variable that changes exactly what you do (such as, grocery shopping is the same routine but the items change).

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

Journal time.

How does a method definition know which values go into each of the variable names?

Come up with a couple of examples from your daily life where you do the same steps, but there is some variable that changes exactly what you do (such as, grocery shopping is the same routine but the items change).

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